Large-Scale Multilingual Document Clustering

A Dissertation
Submitted to the School of Library and Information Science
of Keio University

Kazuaki KISHIDA
May 2013

Abstract

It is often necessary to categorize automatically multilingual document sets, in which documents written in a variety of languages are included, into topically homogeneous subsets, such as when applying an automatic summarization system for multilingual news articles. However, there have been few studies on multilingual document clustering (MLDC) to date. In particular, it is not known whether clustering techniques are effective in large-scale multilingual document sets. When the target multilingual document set is enough small, it is easy to partition it automatically by using machine translation (MT) software and standard statistical package after text processing. In contrast, for other situations where large document collections have to be processed, it is necessary to explore a 'scalable' technique of MLDC because computational complexity of executing MT and DC increases exponentially, not linearly, as size of target data becomes larger. The purpose of this thesis is to develop a method of large-scale MLDC and to verify its effectiveness experimentally. The approach of this thesis for solving the large-scale MLDC problem is to combine cross-lingual information retrieval (CLIR) technique and clustering technique for large-scale document collections. For it, this thesis reviews CLIR methods and document clustering algorithms exhaustively to identify useful techniques for large-scale MLDC in terms of efficiency. As a result, the thesis adopts a combination of dictionary-based translation method in CLIR and leader-follower clustering (LFC) algorithm for implementing the MLDC system. After the reviews, results from three experiments for clarifying empirically the effectiveness of the proposed system are reported, more specifically, (a) effectiveness of scalable techniques for term (translation) disambiguation used in the dictionary-based translation, (b) effectiveness of the LFC algorithm for monolingual DC in comparison with theoretically sophisticated techniques, and (c) effectiveness of the LFC algorithm with dictionary-based translation for large-scale MLDC, by using some test collections. In experiment (a), it was observed that 'best cohesion' method for translation disambiguation works well although its computational complexity is relatively low, and in experiment (b), it was shown that the LFC algorithm for which the target file is scanned only twice can generate 'good' cluster sets, which are comparable with those obtained by the spherical k-means algorithm and the hierarchical Dirichlet process (HDP) mixture model. Finally, it was clarified in experiment (c) that the MLDC system works well for a document collection including over 13,000 news articles written in English, French, German and Italian. Through the experiments, the effectiveness and efficiency of the proposed method for MLDC were empirically confirmed.

Acknowledgements

Findings from my research efforts on cross-lingual information retrieval and document clustering for some years were compiled into this thesis. I am indebted to everyone for providing comments and valuable suggestions during my research activity. Also, I would like to thank my colleagues, most notably Professor Shunsaku Tamura who supervised my work for writing the thesis and Professor Shuichi Ueda who recommended me to submit the thesis. Finally, I am deeply grateful to Professor Masaya Takayama for guiding me to study of library and information science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: Multilingual Document Clustering

1.1 Document Clustering and Information Resource Organization

Suppose that there is a heterogeneous set of documents, which is denoted by $D = \{d_1, d_2, ..., d_N\}$ where N is the total number of documents. If a topic label is assigned to each document for indicating its subject, D can be topically organized based on the labels. For example, in libraries, a classification number is usually allocated to each book by human experts, and a set of various books hold by each library is well-arranged on its shelf by using the classification numbers. This is a fundamental device for organizing a collection of information resources. Actually, the classification numbers allow library users to find books relevant to their information needs on the shelf, or traditional *information retrieval* (IR) systems including OPAC (online public access catalog) systems have often provided a function of utilizing the classification numbers (or comparable 'subject headings') as access points to the databases.

In contrast, when any topic label by human experts is not available for a heterogeneous set of documents, it would be convenient to partition automatically the set into topically homogeneous subsets such that

$$D = C_1 \cup C_2 \cup ... \cup C_L = \{C_k\}_{k=1}^{L}, \tag{1.1}$$

where C_k denotes a *cluster* and L is the total number of clusters [1]. The operation is generally called *document clustering* (DC). More precisely, there are two ways of clustering, namely, exclusive and nonexclusive clustering. In *exclusive clustering*, a document belongs to only a single cluster such $C_k \cap C_h = \emptyset$ where $k \neq h$. Meanwhile, if $C_k \cap C_h \neq \emptyset$, then it is *nonexclusive clustering*. Inevitably, in a situation that a single document tends to discuss multiple topics, it is better to adopt a strategy based on nonexclusive clustering.

Examples of DC are as follows:

- By clustering a huge bibliographic database into some parts beforehand and identifying particular clusters relevant to a given query, more efficient and effective searches of the database may be feasible (i.e., cluster-based IR or distributed IR).
- When an IR system provides search results, it would be convenient for users if a set of documents in the output is divided into some homogeneous groups (this strategy is adopted by some systems, which are often called 'clustering search engines').
- For automatic summarization of news articles in a given period, it is necessary to partition them into subsets reporting individual topics.

[1] C_k is formally defined as a partial set of documents, but it is sometimes used as a label of the subset in this thesis.

- Automatic grouping of electronic documents produced officially in a company may bring discovery of unexpected (novel) topics on the business.

Such kinds of clustering become increasingly an important tool for *text mining* after electronic documents in offices and on the web increment rapidly. Whereas bibliographic control of publications such as books, journal and so on, can be considered as an essential work of librarians, it is unrealistic that human experts as librarians make efforts to organize all electronic resources other than such official publications. So, automatic processing by computers (e.g., automatic grouping of web documents) is indispensable for organizing electronic information resources in terms of efficiency. For the organizing task, DC plays an essential role, and techniques of it have to be enhanced more in order to manage a huge collection of electronic documents effectively and efficiently.

Technically, DC can be regarded as a task of *classification* in which

- objects of classification take a form of text, and
- any answer of classification is not available.

The latter implies that DC is a sort of *unsupervised classification* for which existence of any training set (or answer set) is not assumed. In contrast, *supervised classification* of documents is often called *text categorization*, which utilizes intrinsically different techniques from those for DC, although text categorization and DC have a common factor that the target of analysis is text (i.e., a set of words).

In text categorization, some sophisticated theories and techniques of *machine learning* are available, but it seems that they can not be applied directly to DC problems other than a few exceptions [2]. Unfortunately, it is not possible to use text categorization techniques in a situation that there is no answer and no predefined classification scheme. In such cases, DC is a useful tool for organizing information resources.

1.2 Method of Multilingual Document Clustering

In addition, since multilingual document sets including items written in two or more languages have become more readily available with the spread of the Internet, special algorithms for multilingual document clustering (MLDC) are needed. For example, a system of automatically summarizing multilingual news articles requires effective MLDC for identifying a set of similar news articles (e.g., Chen & Lin(2000) [60], Leftin(2003) [183], Evans & Klavans(2003) [89]). MLDC would be also useful for partitioning patent documents written in multiple languages into topical subgroups for detecting automatically technical trends. Unfortunately, as Wu & Lu(2007) [319] pointed out, there has been little research on MLDC, and it is not sufficiently known which clustering techniques are effective for medium- or large-scale multilingual sets of documents. Researchers have to tackle the problem of large-scale MLDC, which is the main focus of this thesis.

A reason that techniques of large-scale MLDC have not yet been sufficiently explored would be its difficulty as a research task. Basically, for large-scale MLDC, the following two techniques would be indispensable:

- cross-lingual information retrieval (CLIR) technique.
- clustering technique for large-scale document collections.

When the target multilingual document set is enough small, it is easy to divide it automatically by using machine translation (MT) software and standard statistical package after text processing (Section 2.1.6). Actually, there would be many cases in which the combination of MT and statistical software sufficiently works. However, it is necessary to explore a 'scalable' technique of MLDC for other situations where large document collections have to be processed because computational complexity of executing MT and DC increases exponentially, not linearly, as size of target data becomes larger. In particular, DC is a special clustering in that extraordinarily many attributes of

[2] For instance, *semi-supervised classification* tries to learn from both labeled and unlabeled data. The 'labeled data' means a set of documents to which one or more labels (e.g., topic codes) are assigned by human experts as an answer.

the target objects have to be processed as discussed in Section 3.1.1. Thus the system consisting of standard MT and statistical software would not return any result when the document collection size exceeds its limit.

As reviewed in Chapter 3, special algorithms tailored to the large-scale DC problem have been developed so far in the filed of IR, and an optimal one can be selected from the algorithms. Meanwhile, simple and fast conversion of the large document collection from one language to the other language by which search queries are represented has been sometimes attempted in CLIR area, and the technique may be available for translating a large set of documents in DC. The combination of a CLIR technique and a 'scalable' DC algorithm is so promising for large-scale MLDC. This is a main idea of the thesis.

1.3 Purpose and Outline of Thesis

The purpose of this thesis is to develop a method of large-scale MLDC by combining a technique for document translation in CLIR and a scalable DC algorithm developed in IR field. As mentioned above, the combination is so promising for implementing a system which enables to cluster large multilingual document sets, but it has not yet explored in literature. This thesis tries to select appropriate components of the combination through exhaustive literature review and technical discussions on CLIR and DC, and to clarify empirically effectiveness and efficiency of MLDC methods based on the combination by using the Reuters corpora [187]. More specifically, in this thesis, dictionary-based translation method in CLIR and leader-follower clustering (LFC) algorithm are employed for implementing the MLDC system. This method is clearly efficient in terms of computational complexity, and the main purpose of experiments using the Reuters corpora is to confirm 'validity' or 'quality' of clusters generated by the proposed method.

The rest of this thesis is organized as follows. In Chapter 2, standard IR theories and techniques are summarized because DC is largely dependent on IR methods, which will be often used in the successive chapters. Especially, the vector space model in IR plays an important role in DC, and text processing in IR is usually applied to DC. After that, CLIR techniques are exhaustively reviewed in Chapter 2 to identify useful techniques for large-scale MLDC. The CLIR review is an extension of Kishida(2005) [149] and Kishida(2010b) [153].

Next, DC techniques are thoroughly reviewed in Chapters 3 and 4. The review is split into two chapters due to fact that so many DC techniques have been proposed so far. Chapter 3 is an extension of Kishida(2003) [147], which focuses on traditional techniques such as hierarchical clustering, k-means algorithms, leader-follower clustering and so on. In contrast, probabilistic and matrix-based clustering techniques developed mainly after publication of Kishida(2003) [147] are discussed in Chapter 4. After text mining research became active, many novel approaches which can be applied to DC have been proposed one after another in the fields of statistics, machine learning, pattern recognition and so on. They are often based on relatively complicated statistics (e.g., Bayesian model) or matrix algebra, and it is necessary to examine carefully appropriateness of them to DC problems. So, in Chapters 3 and 4, results from simple experiments of DC for the main techniques by using a small sample database are also reported in order to obtain deeper insights on them.

After thorough technical reviews from Chapter 2 to 4, results from three experiments are reported in Chapter 5 for clarifying empirically,

A. Effectiveness of scalable techniques for term (translation) disambiguation used in the dictionary-based translation,

B. Effectiveness of the LFC algorithm for monolingual DC in comparison with theoretically sophisticated techniques, and

C. Effectiveness of the LFC algorithm with dictionary-based translation for large-scale MLDC,

by using some test collections. Experiment A is based on Kishida(2007) [150]. As discussed later, the dictionary-based translation often generates irrelevant translations because entries of bilingual dictionaries usually contain many words with different senses. So, selecting appropriate words from them (i.e., translation disambiguation) is indispensable for CLIR, and many techniques for

it have been developed so far. Among them, this experiment focuses on some 'scalable' methods, and confirms empirically the effectiveness by using a standard IR test collection (CLEF 2003).

Whereas Experiment A is related to the translation module of MLDC, the target of Experiments B is clustering algorithm for MLDC, and so, a monolingual (English) test collection produced for text categorization (Reuters corpora, RCV1) is used in this experiment. Precisely, the experiment consists of two sub-experiments, first of which is to confirm the effectiveness of the LFC algorithm in comparison with a spherical k-means algorithm and hierarchical Dirichlet process (HDP) mixture model which are 'hot' clustering algorithms based on iterative computation. It is self-evident that the LFC algorithm outperforms them in terms of efficiency, and therefore, the effectiveness has to be checked before proceeding to an attempt of MLDC experiment. In contrast, a purpose of the second sub-experiment, which is based on Kishida(2010) [152], is to examine additionally scalability of the LFC algorithm. In the second experiment, larger test collections extracted from the RCV1 corpus are used for measuring the effectiveness of the algorithm.

Finally, an experiment of the MLDC method using a multilingual test collection (Reuters corpora, RCV2) is reported, based on Kishida(2011) [154]. In this experiment, some variations of the MLDC method are examined, and the effectiveness is empirically clarified.

Chapter 2

Technical Review of Cross-Lingual Information Retrieval

2.1 Standard Technique for Information Retrieval

This section reviews key concepts and standard techniques for information retrieval (IR). Understanding them sufficiently is important for exploring not only cross-lingual information retrieval (CLIR) but also document clustering (DC) because DC has a close relationship with IR technically.

2.1.1 Boolean searching

In standard IR situations, documents relevant to an ad-hoc query q given from a user has to be quickly identified from $D = \{d_1, \ldots, d_N\}$. Usually, a document d ($\in D$) is decomposed into a set of terms $\Omega(d) = \{t_1, \ldots, t_m\}$ by text processing, and these terms are respectively registered in an *index file* enabling to locate each term at very high-speed by a search algorithm such as B-tree, hashing and so on (see [280]). The simplest search strategy is to extract documents including all query terms, which formally means that a set of documents to be returned as a search result for q is determined as

$$D_q = \{d \,|\, \Omega(q) \subset \Omega(d)\}, \tag{2.1}$$

where $\Omega(q)$ indicates a set of terms in query q. It is easy to identify D_q quickly by using the index file because a set of documents including term t is recorded in an entry of t in the file. Actually, D_q in Equation (2.1) is calculated as an intersection of these sets,

$$D_q = \bigcap_{t \in \Omega(q)} D_t, \tag{2.2}$$

where D_t is a set of documents including term t [312].

Equation (2.2) indicates that total computational time for finding D_q is approximately proportional to 'addition', not 'multiplication', of each time taken for identifying respective D_t from an index file, which guarantees scalability of the IR system. Although syntactic or semantic information is inevitably lost by reducing a document into a set of terms, this reduction (often referred to as 'bag-of-words architecture') plays a key role so that IR systems work on a very large document set.

Similarly, it may be possible to adopt another strategy,

$$D_q = \{d \,|\, \Omega(q) \cap \Omega(d) \neq \emptyset\}, \tag{2.3}$$

which is operationally computed as

$$D_q = \bigcup_{t \in \Omega(q)} D_t. \tag{2.4}$$

Furthermore, as an natural extension of Equation (2.2) and (2.4), it is feasible to return a search result for a 'structured' query with Boolean operators such as "$q = (t \,\mathrm{OR}\, s) \,\mathrm{AND}\, u$" where s and u are also query terms. For this example, D_q is calculated such that

$$D_q = D_{(t \,\mathrm{OR}\, s) \,\mathrm{AND}\, u} = (D_t \cup D_s) \cap D_u. \tag{2.5}$$

In general, this is called *Boolean searching*. Also, 'NOT' operator is available in addition to 'AND' and 'OR' operators in Equation (2.5) in most of Boolean searching systems [1]. For example, if "$q = t \,\mathrm{NOT}\, s$", then $D_q = D_{t \,\mathrm{NOT}\, s} = D_t \setminus D_s$ where $\setminus$ denotes an operation of a difference set.

2.1.2 Automatic indexing

Tf-idf weighting

In order to enhance search performance, it is crucial to select 'important' terms from those included in each document. For instance, even if a query term appears in a document, the document would be irrelevant unless a topic or a subject implied by the term is sufficiently discussed in it. A conventional method for resolving this problem is to count occurrence frequency of the term in the document, and to assign a weight to the term (i.e., *term weight*) based on it. If the *term frequency* (tf) is higher, then the term is reasonably assumed to be more important in the document [2].

However, it is clearly insufficient to compute the term weight from only the term frequency because general or 'non-specific' terms are often used more frequently in documents and such terms usually do not have ability to discern relevant documents from irrelevant ones. Historically, various methods for automatically identifying 'specific' (not general) terms to be useful for IR had been explored as *automatic indexing* techniques since 1960s (see [172]). Among them, term weighting based on *inverse document frequency* (idf) [288] has been generally used from 1970s until now. For example, it is expected that a term "library" is not useful in a bibliographic database of 'library science' because almost every document contains this term. In contrast, "library" would be able to identify well a specific topic in another database of 'computer science' because this term does not appear ubiquitously in the database. From this example, it is easy to imagine that document frequency (more precisely, the number of documents including a particular term) plays a key role for selecting useful *index terms*. Actually, the effectiveness of idf has been empirically proven through many IR experiments so far.

In order to compute a weight w_{ij} of term t_j in document d_i, tf is usually multiplied by idf. For instance, it becomes that

$$w_{ij} = \mathrm{f}_{ij} \times \log \frac{N}{n_j}, \tag{2.6}$$

where f_{ij} indicates term frequency of t_j in d_i and n_j denotes document frequency of term t_j in D (note that $i = 1, \ldots, N$ and $j = 1, \ldots, M$, i.e., M is the total number of distinct terms in D). Equation (2.6) is a typical *tf-idf weighting*, in which a logarithm [3] is used for calculating the idf factor. The idf factor can be interpreted to measure entropy of the term if n_j/N is considered as

[1] Modern search engines on the Internet would apply 'document ranking' to a set specified by Equation (2.2), basically. When the number of web pages was not so large, there were some engines using the set by Equation (2.4) in the ranking operation for the target query. It should be noted that many ranking algorithms developed in the IR field are implicitly assumed to work on the set by Equation (2.4). The strategy was often referred to as *partial match* because documents not including all query terms can be listed in the search output. In *exact match* strategy, only documents satisfying completely a condition specified by using Boolean operators are included in the output (often without ranking). Generally, when document ranking is applied to set D_q limited by Boolean operators such as Equations (2.2), (2.4) or (2.5), it is often called *best match* strategy.

[2] This idea can be traced back to Luhn(1958) [199].

[3] When $2^x = y$, it is defined that $x = \log_2 y$. Also, if $e^x = y$, then $x = \log_e y$ where e denotes Napier's constant ($e = 2.718\ldots$), which is called 'natural logarithm of y'. In this thesis, the natural logarithm is basically assumed and simply written as $\log y$. As to the logarithm, note that $\log 1 = 0$, $\log(x \times y) = \log x + \log y$, and $\log(x \div y) = \log x - \log y$. In particular, it follows that

$$\log(x_1 \times x_2 \times \ldots \times x_m) = \log \prod_{i=1}^{m} x_i = \sum_{i=1}^{m} \log x_i = \log x_1 + \log x_2 + \ldots + \log x_m,$$

where $\sum_{i=1}^{m} y_i = y_1 + \ldots + y_m$ and $\prod_{i=1}^{m} y_i = y_1 \times \ldots \times y_m$. This mathematical conversion will be often used in successive chapters.

a probability that the term is contained in a document drawn randomly from D [4]. Some other variations of tf-idf weighting have been proposed in the IR field.

Two-Poisson model

In total, m tokens are supposed to be included in a document [5], and a term is assumed to be randomly chosen from a vocabulary with probability p as each token here. If the selection is tried independently each other (i.e., a Bernoulli trial), then the probability $P(x)$ that a particular term selected with a probability p appears x times in a document can be computed by a *binomial distribution*,

$$P(x) = {}_mC_x\, p^x (1-p)^{m-x}, \quad x = 0, 1, 2, \ldots, m, \tag{2.7}$$

where ${}_mC_x = m!/[x!(m-x)!]$ which is called 'binomial coefficient' [6]. If m is so large and p is so small, then the *Poisson distribution*,

$$P(x) = \frac{e^{-\lambda}\lambda^x}{x!}, \quad x = 0, 1, 2, ..., \tag{2.8}$$

is more appropriate where $\lambda = mp$ and λ is a mean of the distribution [7].

Although this model is based on a non-realistic assumption on document writing by human, the Poisson distribution has been often used in the IR or natural language processing (NLP) fields. For instance, in automatic indexing, whether t_j is selected as an index term of d_i or not can be decided based on a probability $P(x \geq \mathrm{f}_{ij})$ following a Poisson distribution with a parameter $\lambda = N^{-1}\sum_{i=1}^{N} \mathrm{f}_{ij}$ (i.e., an average frequency of this term per document) [73]. If the probability is so small (e.g., < 0.05), then term t_j can be interpreted to appear extraordinarily often in document d_i, which leads to selecting it as an index term of this document.

Whereas a single Poisson distribution is possibly enough to describe frequencies of a 'non-specific' term, it may be appropriate for a 'specific' term to use *two-Poisson model* (2P model) [34, 117],

$$P(x) = a\frac{e^{-\lambda}\lambda^x}{x!} + (1-a)\frac{e^{-\lambda'}\lambda'^x}{x!}, \tag{2.9}$$

where λ is an average frequency of the specific term within a set of documents relevant to the subject implied by the term and λ' is one among irrelevant documents (a is a mixing parameter, i.e., $0 < a < 1$). If the two distributions are completely separated with a boundary $x = b$, then it is enough to take a simple strategy that term t_j is teated as an index term for only documents in which its frequency f_{ij} is over b (i.e., $\mathrm{f}_{ij} > b$). In contrast, when two distributions are crossing each other (e.g., see Figure 4.1 in Chapter 4), a conditional probability that document d_i belongs to a set of documents relevant to t_j,

$$P(d_i \in D_{\mathcal{R}} | x = \mathrm{f}_{ij}) = \frac{P(d_i \in D_{\mathcal{R}}, x = \mathrm{f}_{ij})}{P(x = \mathrm{f}_{ij})} = \frac{ae^{-\lambda}\lambda^{\mathrm{f}_{ij}}}{ae^{-\lambda}\lambda^{\mathrm{f}_{ij}} + (1-a)e^{-\lambda'}\lambda'^{\mathrm{f}_{ij}}}, \tag{2.10}$$

is computed based on Equation (2.9) where $D_{\mathcal{R}}$ indicates the set of relevant documents [8], and then, t_j is adopted as an index term if its probability is over a threshold [118].

---

[4] If n_j/N is regarded as a probability that a document drawn randomly from the entire set includes term t_j, then $\log(N/n_j)$ could be considered as 'entropy' of t_j. Basically, entropy is defined as $\log p^{-1} = -\log p$ where p is a probability that an event occurs (in this case, $p = n_j/N$).

[5] For example, a sentence "This apple is that apple" includes two tokens of word "apple" and $m = 5$.

[6] Note that $x! = x \times (x-1) \times (x-2) \times \ldots \times 3 \times 2 \times 1$. The binomial coefficient represents the number of combinations when x objects are selected from m objects ($x \leq m$). If $m = 5$, $x = 3$ and $p = 1/4$, then $P(3) = [5!/(3!2!)](1/4)^3(3/4)^2$. Or, when $m = 5$, $x = 5$ and $p = 1/4$, $P(5) = [5!/(5!0!)](1/4)^5(3/4)^0 = 1 \times (1/4)^5 \times 1$ because $x^0 = 1$ and $0! = 1$ by its definition.

[7] In general, $x^{-y} = 1/x^y$, and specially, $x^{-1} = 1/x$ and $x^{-1/2} = 1/\sqrt{x}$. As to exponential computation, note that $x^y \times x^z = x^{y+z}$ and $x^y \div x^z = x^{y-z}$, and $(xy)^z = x^z \times y^z$. The Poisson distribution can be obtained by a limit of binomial distribution when $p \to 0$ and $m \to \infty$ under a condition that $mp = \lambda =$ constant.

[8] Conditional probability $P(y|x)$ is defined such that $P(y|x) = P(x,y)/P(x)$ where $P(x,y)$ denotes a 'joint probability' of random variables x and y. Also, in this case, $P(x)$ is 'marginal probability' of $P(x,y)$ and $P(x) = \sum_y P(x,y)$ if x and y are discrete variables. Note that $\sum_y$ means the sum of $P(x,y)$ for all possible values of y here.

2.1.3 Vector space model

Unlike traditional IR systems such as conventional OPAC systems, prototype systems developed by IR researchers and search engines on the Internet usually try to rank documents in D_q of Equations (2.2), (2.4) or (2.5) in decreasing order of relevance to a given query. The *ranked output* would reduce information overload of users in checking search results if the degree of relevance is correctly estimated by IR systems. The degree is often called '*document score*' or '*retrieval status value* (RSV)'[33], which is computed based on weights of terms contained in each document. The document score is denoted by v_i where index i indicates document d_i.

Vector as subject representation

In *vector space model* [270, 269], which is one of the earliest IR models for ranked output [9], documents and queries are algebraically expressed as vectors whose elements are term weights such as Equation (2.6), and each document score for a given query is calculated as cosine of the angle between vectors of them (see Figure 3.1 in Chapter 3). A vector of document d_i $(i = 1, \ldots, N)$ can be written as

$$\mathbf{d}_i = [w_{i1}, w_{i2}, \ldots, w_{iM}]^T, \tag{2.11}$$

and similarly, the query vector is constituted as $\mathbf{q} = [w_{q1}, \ldots, w_{qM}]^T$ where w_{qj} is a weight of term t_j in the query $(j = 1, \ldots, M)$ [10]. The vector is interpreted as a *subject representation* of documents or queries, and topical similarity between two representations is measured as a *cosine coefficient*. For instance, the similarity of representation between document d_i and query q, which is denoted by $s(\mathbf{d}_i, \mathbf{q})$, can be computed as

$$s(\mathbf{d}_i, \mathbf{q}) = \frac{\mathbf{d}_i^T \mathbf{q}}{\|\mathbf{d}_i\| \, \|\mathbf{q}\|} = \frac{\sum_{j=1}^{M} w_{ij} w_{qj}}{\sqrt{\sum_{j=1}^{M} w_{ij}^2} \sqrt{\sum_{j=1}^{M} w_{qj}^2}}, \tag{2.12}$$

where $0 \leq s(\mathbf{d}_i, \mathbf{q}) \leq 1$ if all term weights are non-negative. The value of $s(\mathbf{d}_i, \mathbf{q})$ can be reasonably used as a d_i's document score v_i for ranked output.

In Equation (2.12), an inner product of two vectors is divided by a product of norms of them, which means that the length of both vectors is normalized [11]. The normalization of vector length is expected to work for preventing that similarity values of longer documents are overestimated regardless of their relatedness to the query.

Term weighting in vector space model

The formula of cosine coefficient in Equation (2.12) is basically decomposed into three parts: (a) term weighting in document, w_{ij}, (b) term weighting in query, w_{qj}, and (c) normalization [267]. For instance, a simple strategy is to set the parameters such as

[9] Vector space model had been constructed by a research group of G. Salton in the process of developing SMART[266] since 1961, which was an advanced IR system at that time.

[10] For instance, vector $\mathbf{x}$ can be defined by putting two real numbers, x_1 and x_2, into a column such that

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = [x_1, x_2]^T,$$

where T means transposition of columns and rows (i.e., $\mathbf{x}$ is a column vector and $\mathbf{x}^T$ is a row vector). In particular, $\mathbf{x}$ is a two-dimensional vector because two numbers are placed in the column. Addition and subtraction of two vectors are computed as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \end{bmatrix}, \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} x_1 - y_1 \\ x_2 - y_2 \end{bmatrix},$$

respectively. Also multiplication of two vectors is defined such that

$$\mathbf{x}^T \mathbf{y} = [x_1, x_2] \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = x_1 y_1 + x_2 y_2, \quad \mathbf{x}\mathbf{y}^T = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} [y_1, y_2] = \begin{bmatrix} x_1 y_1 & x_1 y_2 \\ x_2 y_1 & x_2 y_2 \end{bmatrix}.$$

The former $\mathbf{x}^T \mathbf{y}$ is called '*inner product*' and its computational result becomes a single number. In contrast, outcome of $\mathbf{x}\mathbf{y}^T$ is a matrix with two columns and two rows (i.e., a 2×2 matrix). If k is a number (which is often called 'scalar'), then simply $k\mathbf{x}^T = k[x_1, x_2]^T = [kx_1, kx_2]^T$.

[11] Namely, the norm can be interpreted as a length of the vector, which is defined as $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T \mathbf{x}}$.

- $w_{ij} = \mathrm{f}_{ij} \times \log N/n_j$,
- $w_{qj} = \mathrm{f}_{qj} \times \log N/n_j$,
- normalization by $\sqrt{\sum_{j=1}^{M} w_{ij}^2} \times \sqrt{\sum_{j=1}^{M} w_{qj}^2}$,

where f_{qj} denotes a frequency of term t_j in the query. Otherwise, as a variation, an inner product of two vectors whose elements are

$$w_{ij} = \frac{(\log \mathrm{f}_{ij} + 1.0) \times \log(N/n_j)}{\sqrt{\sum_{j=1}^{M} [\log(\mathrm{f}_{ij} + 1.0) \times \log(N/n_j)]^2}}, \tag{2.13}$$

and

$$w_{qj} = \left(0.5 + 0.5 \frac{\mathrm{f}_{qj}}{\max_{j=1,\ldots,M} \mathrm{f}_{qj}}\right) \times \log \frac{N}{n_j}, \tag{2.14}$$

respectively, may be used as $s(\mathbf{d}_i, \mathbf{q})$ with no more explicit normalization [43] because normalizing factors are already included in definitions of w_{ij} and w_{qj} (strictly, if $\mathrm{f}_{ij} = 0$, then w_{ij} is assumed to be zero in Equation (2.13)). As shown in these instances, many variations for computing the similarity measure can be explored in the framework of vector space model [267, 113].

Pivoted document length normalization

In a situation that longer documents tend to be more relevant, the normalization by norms would not work positively, and it may be better to reduce the normalizing factor (i.e., $\|\mathbf{d}_i\| \times \|\mathbf{q}\|$) for longer documents and to increase it for shorter ones. In the *pivoted document length normalization* scheme [285], the document score is computed such that

$$\mathrm{v}_i = s(\mathbf{d}_i, \mathbf{q}) = \frac{\sum_{j=1}^{M} w_{qj} w_{ij}}{(1.0 - s_l) p_v + s_l \tilde{w}_i}, \tag{2.15}$$

where $\tilde{w}_i = (\sum_{j=1}^{M} w_{ij}^2)^{1/2} \times (\sum_{j=1}^{M} w_{qj}^2)^{1/2}$, p_v indicates a 'pivot' used for changing values of the normalizing factor between shorter and longer documents (e.g., an average of $\tilde{w}_i$ is used as p_v such that $p_v = N^{-1} \sum_{i=1}^{N} \tilde{w}_i$), and s_l is a constant. If $s_l < 1.0$, then the normalizing factor of longer document d_i where $\tilde{w}_i > p_v$ is proportionally decreasing (note that the denominator of Equation (2.15) is converted into $p_v + s_l(\tilde{w}_i - p_v)$), and it is expected to avoid underestimating score of longer documents to be relevant.

2.1.4 Probabilistic IR model

In *probabilistic IR model* which is another prevailing model for ranked output, each document scores is theoretically estimated based on a quantity $P(\mathcal{R}|d_i, q)$, which means a conditional probability of relevance (denoted by $\mathcal{R}$) when document d_i and query q are given [12]. Since d_i is usually replaced by vector representation $\mathbf{d}_i$ in the same manner as vector space model, the probability becomes $P(\mathcal{R}|\mathbf{d}_i, q)$ [256].

Classical binary independent model

According to Bayes' formula [13],

$$P(\mathcal{R}|\mathbf{d}_i, q) = \frac{P(\mathbf{d}_i|\mathcal{R}, q) P(\mathcal{R}|q)}{P(\mathbf{d}_i|\mathcal{R}, q) P(\mathcal{R}|q) + P(\mathbf{d}_i|\bar{\mathcal{R}}, q) P(\bar{\mathcal{R}}|q)}, \tag{2.16}$$

[12] If $x = 1$ means an event that term t_1 appears in a document and $y = 1$ indicates another event that term t_2 appears in a document, then $P(x = 1, y = 1)$ corresponds to the probability that both terms appear in a document. Thus, $P(x = 1|y = 1)$ indicates the probability that a document in which t_2 appears includes t_1.

[13] Bayes' formula can be derived from the definition of conditional probability, $P(x|y) = P(x, y)/P(y) = P(y|x)P(x)/P(y)$, which gives a relation between $P(x|y)$ and $P(y|x)$.

Table 2.1: Numbers of relevant and irrelevant documents for a query

	relevant	irrelevant	Total
t_j appears	r_j	$n_j - r_j$	n_j
t_j does not appear	$R - r_j$	$N - R - n_j + r_j$	$N - n_j$
Total	R	$N - R$	N

where $\bar{\mathcal{R}}$ indicates irrelevance, and its logit conversion $\log[P/(1-P)]$ becomes

$$O(\mathcal{R}|\mathbf{d}_i, q) = \log\left[\frac{P(\mathbf{d}_i|\mathcal{R}, q)P(\mathcal{R}|q)}{P(\mathbf{d}_i|\bar{\mathcal{R}}, q)P(\bar{\mathcal{R}}|q)}\right] = \log\left[\frac{P(\mathbf{d}_i|\mathcal{R}, q)}{P(\mathbf{d}_i|\bar{\mathcal{R}}, q)}\right] + C_1 \tag{2.17}$$

where $C_1 = \log[P(\mathcal{R}|q)/P(\bar{\mathcal{R}}|q)]$ (i.e., C_1 is constant for all documents).

If binary document vector such that $\mathbf{d}_i = [0, 1, 1, \ldots, 0]^T$ in which '1' indicates that the corresponding term appears in the document and '0' does not is adopted, and term independence in computing $P(\mathbf{d}_i|\mathcal{R}, q)$ is assumed [14], then it becomes

$$P(\mathbf{d}_i|\mathcal{R}, q) = \prod_{j=1}^{M} P(t_j|\mathcal{R}, q)^{x_{ij}}[1 - P(t_j|\mathcal{R}, q)]^{1-x_{ij}}, \tag{2.18}$$

where $P(t_j|\mathcal{R}, q)$ is a probability that t_j appears in documents relevant to q, and x_{ij} indicates j-th element in the binary vector of d_i (i.e., $x_{ij} = 1$ or $x_{ij} = 0$). Similarly, $P(\mathbf{d}_i|\bar{\mathcal{R}}, q)$ is also represented in the same form as Equation (2.18) by replacing $P(t_j|\mathcal{R}, q)$ with $P(t_j|\bar{\mathcal{R}}, q)$. Therefore, finally, Equation (2.17) becomes

$$O(\mathcal{R}|\mathbf{d}_i, q) = \sum_{j=1}^{M} x_{ij} \log \frac{P(t_j|\mathcal{R}, q)[1 - P(t_j|\bar{\mathcal{R}}, q)]}{[1 - P(t_j|\mathcal{R}, q)]P(t_j|\bar{\mathcal{R}}, q)} + C_1 + C_2, \tag{2.19}$$

where $C_2 = \log \sum_{j=1}^{M}[1 - P(t_j|\mathcal{R}, q)]/[1 - P(t_j|\bar{\mathcal{R}}, q)]$ that is also constant for all documents.

It is easy to estimate probabilities in Equation (2.19) by using term occurrence statistics shown in Table 2.1 such that

$$\hat{P}(t_j|\mathcal{R}, q) = r_j/R, \tag{2.20}$$

$$\hat{P}(t_j|\bar{\mathcal{R}}, q) = (n_j - r_j)/(N - R), \tag{2.21}$$

where r_j is the number of relevant documents including t_j and R is the total number of relevant documents. By substituting them into Equation (2.19) and neglecting C_1 and C_2, it becomes

$$O(\mathcal{R}|\mathbf{d}_i, q) \simeq \tilde{O}(\mathcal{R}|\mathbf{d}_i, q) = \sum_{j=1}^{M} x_{ij} \log \frac{r_j(N - R - n_j + r_j)}{(R - r_j)(n_j - r_j)}, \tag{2.22}$$

which is the classical *binary independent model* [257]. If statistics in Table 2.1 can be estimated, then it is possible to use $\tilde{O}(\mathcal{R}|\mathbf{d}_i, q)$ in Equation (2.22) as document score v_i.

If putting that $u_j = \log[r_j(N - R - n_j + r_j)]/[(R - r_j)(n_j - r_j)]$, then Equation (2.22) is simply written as $\sum_{j=1}^{M} x_{ij}u_j$. Table 2.1 is a kind of 2×2 contingency table, and therefore, 0.5 should be added in each cell as a correction for a small sample [254], which leads to

$$u_j = \log \frac{(r_j + 0.5)(N - R - n_j + r_j + 0.5)}{(R - r_j + 0.5)(n_j - r_j + 0.5)}. \tag{2.23}$$

Equation (2.23) has been often used as a formula for term weighting in IR.

[14] If x is probabilistically independent of y, then it follows that $P(x|y) = P(x)$. By definition of conditional probability, a joint probability $P(x, y, z)$ is transformed such that $P(x, y, z) = P(x, y|z)P(z) = P(x|y, z)P(y|z)P(z)$. Thus if the three variables are independent each other, then $P(x, y, z) = P(x)P(y)P(z)$. More generally, $P(x_1, \ldots, x_m) = \prod_{j=1}^{m} P(x_j)$ if all variables are independent.

Okapi formula

A method for removing the 'binary assumption' in Equation (2.22) is to replace binary variable x_{ij} by functions of term frequencies, $g(\mathrm{f}_{ij})$ and $g(\mathrm{f}_{qj})$. Thus it becomes

$$\tilde{O}(\mathcal{R}|\mathbf{d}_i, q) = \sum_{j=1}^{M} g(\mathrm{f}_{ij}) g(\mathrm{f}_{qj}) \log \frac{(r_j + 0.5)(N - R - n_j + r_j + 0.5)}{(R - r_j + 0.5)(n_j - r_j + 0.5)} \tag{2.24}$$

(note that Equation (2.23) is used as u_j in this equation). For instance, in the Okapi system [258, 259], it is supposed that

$$g(\mathrm{f}_{ij}) = \log \frac{P(\mathrm{f}_{ij}|\mathcal{R}, q)/P(\mathrm{f}_{ij} = 0|\mathcal{R}, q)}{P(\mathrm{f}_{ij}|\bar{\mathcal{R}}, q)/P(\mathrm{f}_{ij} = 0|\bar{\mathcal{R}}, q)}, \tag{2.25}$$

and a two-Poisson model which is more complicated than Equation (2.10) is used for describing $P(\cdot)$ in this equation. Because the resulting function $g(\mathrm{f}_{ij}) \times g(\mathrm{f}_{qj})$ becomes too complicated, some approximations are introduced for computation of $\tilde{O}(\mathcal{R}|\mathbf{d}_i, q)$ such that

$$g(\mathrm{f}_{ij}) \approx c_1 \frac{\mathrm{f}_{ij}}{k_1[(1 - b) + b(l_i/\bar{l})] + \mathrm{f}_{ij}}, \tag{2.26}$$

and,

$$g(\mathrm{f}_{qj}) \approx c_3 \frac{\mathrm{f}_{qj}}{k_3 + \mathrm{f}_{qj}}, \tag{2.27}$$

where l_i indicates *document length* of d_i (i.e., $l_i = \sum_{j=1}^{M} \mathrm{f}_{ij}$), $\bar{l} = N^{-1} \sum_{i=1}^{N} l_i$, and b, c_1, c_3, k_1 and k_3 are parameters.

If a set of parameters such that

$$c_1 = k_1 + 1,\ c_3 = k_3 + 1,\ k_1 = 2.0,\ k_3 = \infty,\ b = 0.75,$$

are adopted and it is assumed that $R = r_j = 0$ when any relevance assessment is not made by users, then Equation (2.24) becomes

$$\tilde{O}(\mathcal{R}|\mathbf{d}_i, q) = \sum_{j=1}^{M} \frac{3.0\,\mathrm{f}_{ij}}{0.5 + 1.5(l_i/\bar{l}) + \mathrm{f}_{ij}} \times \mathrm{f}_{qj} \times \log \frac{N - n_j + 0.5}{n_j + 0.5}, \tag{2.28}$$

which is called *Okapi BM25 formula* [259]. In Equation (2.28), it is possible to interpret $\log(N - n_j + 0.5)/(n_j + 0.5)$ as an idf factor and also $1.5(l_i/\bar{l})$ as a normalizing factor. This implies that the Okapi formula consists of tf-idf weights and document length normalization in the same way as vector space model.

Language modeling for IR

Generally, *language model* means a probabilistic distribution of term occurrence in a language. If a probability of generating a set of query terms from a language model derived from each document can be computed, then it is possible to use it as a document score for ranked output in IR [126, 239] such as $\mathrm{v}_i = P(\Omega_q|\mathcal{M}_i)$ where $\mathcal{M}_i$ denotes a language model estimated from d_i. Furthermore, an assumption of term independence enables to write it as $P(\Omega_q|\mathcal{M}_i) = \prod_{j:t_j \in \Omega_q} P(t_j|\mathcal{M}_i)$.

Typically, a formula,

$$P(t_j|\mathcal{M}_i) = aP(t_j|d_i) + (1 - a)P(t_j), \tag{2.29}$$

is often used where $P(t_j|d_i)$ means an empirical probability that term t_j appears in document d_i, $P(t_j)$ indicates a general probability that term t_j is used in documents, and a is a mixing parameter ($0 \leq a \leq 1$). Note that $P(t_j)$ in Equation (2.29) prevents that $P(\Omega_q|\mathcal{M}_i)$ becomes always zero unless all query terms appear in the document (i.e., so-called 'zero-frequency problem').

When $P(t_j|d_i)$ is estimated as f_{ij}/l_i and $P(t_j)$ as n_j/N, $P(\Omega_q|\mathcal{M}_i)$ can be written as

$$P(\Omega_q|\mathcal{M}_i) = \prod_{j:t_j \in \Omega_q} a\frac{\mathrm{f}_{ij}}{l_i} + (1 - a)\frac{n_j}{N}. \tag{2.30}$$

By multiplying $\epsilon = \prod_{j:t_j \in \Omega_q} (an_j)^{-1}$ to both the sides of Equation (2.30),

$$\epsilon P(\Omega_q|\mathcal{M}_i) = \prod_{j:t_j \in \Omega_q} \frac{a}{an_j} \frac{\mathrm{f}_{ij}}{l_i} + \frac{(1-a)}{an_j} \frac{n_j}{N} = \prod_{j:t_j \in \Omega_q} \left[\frac{1}{n_j} \frac{\mathrm{f}_{ij}}{l_i} + \frac{(1-a)}{aN} \right], \tag{2.31}$$

is obtained, which means that this formula contains tf-idf weights and document length normalization like other IR models [126] (note that ϵ is constant for all documents).

Equation (2.29) is generally called *Jelinek-Mercer smoothing*. Meanwhile, in the case of Bayesian smoothing using the Dirichlet distribution as a prior,

$$P(t_j|\mathcal{M}_i) = \frac{\mathrm{f}_{ij} + \eta P(t_j)}{\sum_{j=1}^{M} \mathrm{f}_{ij} + \eta} \tag{2.32}$$

is used where η is a predefined parameter (e.g., $\eta = 2000$) [333]. In general, other smoothing techniques have been developed, and it may be possible to apply them for ranking documents in IR.

Other probabilistic IR models

In addition, there are many probabilistic IR models such as the logistic regression model [104], the Bayesian inference network [47], the DFR (divergence from randomness) model [8], and so on.

2.1.5 Query expansion

IR models for ranking documents would not work well in real situations unless *information needs* from users are correctly and sufficiently represented as search requests or queries to the IR system. However, it is not easy for actual users to come up with appropriate terms representing their needs because usually the users try to search for what they do not know enough. Actually, transaction log analyses using data from operational search engines have revealed that users enter only around two words as search terms on average (e.g, see [204] for exact statistics). In order to alleviate the short query problem, *query expansion* techniques have been explored for adding relevant terms automatically or semi-automatically to original short queries from users.

Thesaurus-based expansion

In practical IR services, thesaurus-based expansion has been manually attempted by users or experts (e.g., librarians). One of the most famous thesauri would be the MeSH (Medical Subject Headings), which provides a controlled vocabulary including synonyms and related terms (RTs) of each medical concept as well as its hypernyms (broader terms: BTs) and hyponyms (narrower terms: NTs). It is possible to enrich original queries by referring to such kind of thesauri (e.g, see [192] using the WordNet as a thesaurus for query expansion).

When synonymous relationships are formally represented as an $M \times M$ matrix

$$\mathbf{M} = [\tau_{jk}], \quad j,k = 1,\ldots,M, \tag{2.33}$$

where $\tau_{jk} = 1$ if term t_j is a synonym of term t_k and otherwise $\tau_{jk} = 0$, original query $\mathbf{q}$ can be converted as $\mathbf{q}' = \mathbf{Mq}$ [269, 243] [15]. If j-th element of $\mathbf{q}$ is defined simply as f_{qj} (i.e., $w_{qj} = \mathrm{f}_{qj}$), then f_{qj} is added to k-th element of new query vector $\mathbf{q}'$ when $\tau_{jk} = 1$. In the case that t_k is not included in the original query, this means that synonymous term t_k is newly added.

[15] More specifically, if $M = 3$ is assumed, then

$$\mathbf{q}' = \mathbf{Mq} = \begin{bmatrix} \tau_{11} & \tau_{12} & \tau_{13} \\ \tau_{21} & \tau_{22} & \tau_{23} \\ \tau_{31} & \tau_{32} & \tau_{33} \end{bmatrix} \begin{bmatrix} w_{q1} \\ w_{q2} \\ w_{q3} \end{bmatrix} = \begin{bmatrix} w'_{q1} \\ w'_{q2} \\ w'_{q3} \end{bmatrix}.$$

An $n \times m$ matrix can be constituted by juxtaposing m n-dimensional vectors. Multiplication of two matrices (including vectors as an $n \times 1$ matrix) such as $\mathbf{AB} = \mathbf{C}$ is feasible only when the number of columns in $\mathbf{A}$ is equal to the number of rows in $\mathbf{B}$. Namely, if $\mathbf{A}$ is an $n \times m$ matrix and $\mathbf{B}$ is an $m \times k$ matrix, then the multiplication is allowed and the resulting $\mathbf{C}$ becomes an $n \times k$ matrix. In this case, (i,j)-element of $\mathbf{C}$ is computed as an inner

Otherwise, it is feasible to define τ_{jk} as

$$\tau_{jk} = \frac{\sum_{i=1}^{N} w_{ij} w_{ik}}{\sqrt{\sum_{i=1}^{N} w_{ij}^2} \sqrt{\sum_{i=1}^{N} w_{ik}^2}}, \tag{2.34}$$

where w_{ij} and w_{ik} are term weights like that in Equation (2.6) [292]. It should be noted that τ_{jk} in Equation (2.34) is computed as a cosine measure between two N-dimensional term vectors whose i-th element is a term weight in document d_i according to the vector space model. The matrix $\mathbf{M} = [\tau_{jk}]$ based on Equation (2.34) is often called statistical thesaurus or *similarity thesaurus*, which can be automatically constructed from a corpus unlike the MeSH and other standard thesauri [16].

Relevance feedback

If some relevant documents are known a priori, then appropriate terms not included in the original query can be extracted from them. Also, it is possible to get such relevant documents by asking actual user to select those from a result obtained by searching for the original query. This is called *relevance feedback*. A well-known term re-weighting technique based on relevance feedback is *Rocchio's method* [260], in which the original query is modified such as

$$\mathbf{q}' = \alpha \mathbf{q} + \frac{\beta}{|D_{\mathcal{R}}|} \sum_{i:d_i \in D_{\mathcal{R}}} \mathbf{d}_i - \frac{\gamma}{|D_{\bar{\mathcal{R}}}|} \sum_{i:d_i \in D_{\bar{\mathcal{R}}}} \mathbf{d}_i, \tag{2.35}$$

where $D_{\mathcal{R}}$ denotes a set of relevant documents, $D_{\bar{\mathcal{R}}}$ is a set of irrelevant documents, and α, β and γ are parameters [268] (e.g., $\alpha = 16$, $\beta = 8$ and $\gamma = 4$) [17]. As shown in Equation (2.35), weights of terms that tend to be included in relevant documents become larger and vice versa, and in this framework, not only new terms may be added but also weights of original query terms may be changed.

Pseudo-relevance feedback (PRF) is a technique of using the relevance feedback with no relevance information by supposing compulsorily that top-ranked documents are relevant. Namely, an initial search run is executed by using an original query, and then, the second run uses a new query obtained by a relevance feedback technique under an assumption that top-ranked documents (e.g., up to 10th rank) are relevant.

In order to extract useful terms from the top-ranked documents, the term weight in Equation (2.23) is often used. More specifically, after ranking terms included in $D_{\mathcal{R}}$ by weight such that

$$\omega_j = r_j \times u_j = r_j \times \log \frac{(r_j + 0.5)(N - R - n_j + r_j + 0.5)}{(R - r_j + 0.5)(n_j - r_j + 0.5)}, \tag{2.36}$$

top-ranked terms (e.g., up to 30th rank) can be considered as new search terms to be added to the original query (it should be noted that $R = 10$ if ten top-ranked documents are assumed to be relevant). When original query terms are included in the list of top-ranked terms, weights of the terms are usually changed (i.e., *re-weighting*). An example of heuristic rule for re-weighting is to change weight w_{qj} into $1.5 \times w_{qj}$ for original query terms and to set $w_{qj} = 0.5$ for newly added terms.

From results of many IR experiments, it is empirically known that PRF tends to improve performance in comparison with its initial searching. Whereas thesaurus-based expansion reflects a macro trend in a topic area or in a corpus regardless of context of each query, the PRF tries to

---

product of i-th row of $\mathbf{A}$ and j-th column of $\mathbf{B}$. For instance,

$$\mathbf{AB} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{(1)}^T \\ \mathbf{a}_{(2)}^T \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{(1)}^T \mathbf{b}_1 & \mathbf{a}_{(1)}^T \mathbf{b}_2 \\ \mathbf{a}_{(2)}^T \mathbf{b}_1 & \mathbf{a}_{(2)}^T \mathbf{b}_2 \end{bmatrix},$$

where $\mathbf{a}_{(i)}$ denotes i-th column vector of matrix $\mathbf{A}^T$. Note that $[\mathbf{AB}]^T = \mathbf{B}^T \mathbf{A}^T$.

[16] It should be noted that some researchers have suspected effectiveness of applying similarity thesaurus derived from term co-occurrence statistics [185, 231].

[17] When A is a set, $|A|$ denotes the number of elements included in A.

extract new terms from a partial set of documents that are expected to be relevant to it. Because the result of macro-level analysis is not always appropriate to a given query, the PRF based on micro-level analysis would be better.

Of course, the PRF is not a perfect tool. For instance, when almost no 'true' relevant document is included in the set of top-ranked documents, the PRF would inevitably deteriorate the performance. In general, it is difficult to predict automatically whether PRF has a positive effect for a given query or not.

2.1.6 Text processing

Word segmentation

In order to decompose each document into a set of terms and to register them into an index file, *word segmentation* (or *tokenization*) has to be executed beforehand as an operation of text processing. It is relatively easy to automatically identify each word from the target text in the case that a space is explicitly inserted between words like in English. For example, "Introduction to Database Management Systems" is naturally decomposed into "Introduction/to/Database/Management/Systems" by checking each space.

After a functional word such as "to" was removed as a *stopword*, remaining words are adopted as index terms (they are sometimes called 'content-bearing words'). Usually, before these terms are sent to the index file, case normalization is applied (e.g., changing "Database" to "database") for preventing failure of string matching in the step of searching the index file.

As another normalization for string matching on the index file, IR systems often incorporate automatic *stemming*, by which suffix is removed and a stem is extracted from a word. In the above example, "systems" would be converted to "system" by a stemming algorithm (or stemmer) [18]. Although harmful words are sometimes yielded by over-stemming (e.g., in the case that "factory" is converted to "factor"), it is expected that the stemming reduces search failure of relevant documents. Also, for a similar objective, *lemmatization* for normalizing derivative words (e.g., "are") into a primitive form (e.g., "be") is sometimes used in some languages.

In the case of languages in which there is no explicit word boundary (e.g., Japanese or Chinese), it is necessary to apply a more complicated algorithm for determining each index term. A basic technique is to extract a character string identical to an entry term of a *machine-readable dictionary* (MRD). Suppose that a text is "abcdefghijk", and "cde" and "hijk" are registered in an MRD as entries. By matching the text with each headword of entries in the MRD, it is feasible to break down the text such as "ab/cde/fg/hijk", where "ab" and "fg" are not registered in the MRD (i.e., unknown words) [19]. Also, a *morphological analyzer* developed for *natural language processing* (NLP) can be employed, if available. For example, Japanese morphological analyzers provide usually two functions, word segmentation and *part-of-speech* (POS) tagging, which enable to indexing Japanese text for IR systems [20].

Phrase identification

In order to enhance search effectiveness, it is important to identify phrases or compound words in a text. For example, if a query "hot dog" is simply decomposed into "hot" and "dog", then the search becomes equivalent to a simple Boolean formula "hot AND dog" (or "hot OR dog"), which possibly yields many irrelevant documents.

To solve this problem, it may be necessary to adopt another search algorithm by which a character string can be located as it is in the text, not based on explicit extraction of index terms. For example, the phrase search function equipped in a search engine allows to find web pages including a representation "I have an apple in my hand" as it is. One of the algorithms for enabling to execute such searches is *suffix tree* (see [280]).

[18] In IR, *Porter's stemming algorithm* [240] is widely used for some languages. Its source code can be downloaded from a web site of the Snowball project by Porter (see Chapter 5).

[19] This technique can be applied to decomposition of compound words in German and other languages (see Sections 5.2 and 5.4).

[20] Another powerful technique is to extract automatically character-based overlapped *n-grams* from Japanese or Chinese text instead of identifying correctly words. In the case of $n = 2$, they are called bi-grams, and when $n = 1$, uni-grams. However, for executing CLIR, it would be necessary to identify words linguistically in most of cases.

In the case of using conventional index files, phrases or compound words have to be found in the process of indexing. If a dictionary of phrases or compound words is available, then they can be detected by matching operation with entry terms of the dictionary. Another simple method without such a dictionary is to extract automatically word-based n-grams from text and to use them as index terms. For example, word-based bi-grams in a text "database management systems" are "database management" and "management systems" and its tri-gram becomes "database management systems" as it is. Otherwise it may be possible to apply POS tagging for identifying phrases, namely, 'noun-noun' or 'adjective-noun' sequences of words would be reasonably assumed to be phrasal representations.

Proximity operators

If positions at which each word is appearing in a document are recorded in an index file, then more flexible phrase searching is possible by using the positional information. Suppose that there are two character strings, (a) "A medical library ..." and (b) "A library of medical science ...". In (a), the positions of "medical" and "library" are 2nd and 3rd, respectively, and in (b) 4th and 2nd. A simple calculation $2 - 3 = -1$ means that the two words are adjacent in this order and $4 - 2 = +2$ indicates that the two words appear in the reverse order with an interval word. This information may allow users to specify a range and the order of occurrence of search terms so as to cover some variations of a phrasal representation. Actually, some IR systems have provided '*proximity operator*s' for this purpose (see [142] for details).

2.1.7 Relevance and its assessment

The objective of IR is to find documents relevant to users' *information needs*. However, the *relevance* is not a simple concept, which has historically led to many different interpretations [272]. For obtaining an overview of them, it is convenient to consider an IR process as a series of four elements, (a) problem situation, (b) information need, (c) search request and (d) query. An information need is derived from a *problem situation* in which the user has an unsolved question. For solving it, the information need is translated into a *search request*, or *query* that IR systems can understand, and actual searches are executed based on them. In general, information needs are assumed to be existing in mind of each user and to be not represented in any verbal form whereas a search request is verbal manifestation of the information need. Also, a query can be interpreted as a special search request tailored for a specific IR system. Search requests have to be produced by end-users to communicate with reference librarians or IR specialists (i.e., searchers) in IR services.

The degree to which the subject of a document is coincident with the search request or query is related to a concept of *topicality*, which can be considered as 'objective' relevance because someone rather than the user having the information need can assess topicality of a document based on the written (or spoken) statement as a search request or query. In contrast, a relationship between the information need and the subject of a document is often referred to as *pertinence*. Since the information need can be recognized only by the user, the pertinence is considered as 'subjective'.

In library and information science, 'situationality' and dynamic nature of the relevance concept as well as its subjectivity are often emphasized by the researchers adopting user-oriented approaches (see [277]). For instance, the degree to which a document is relevant varies as the problem situation is changed with time even if the same user always judges the degree of relevance.

Furthermore, objectively or subjectively relevant documents sometimes are assessed according to the following concepts (see [271, 272, 273]):

- *Novelty*. Whether the information brought by the document is already known for the user or not.
- *Understandability*. Whether the document is readable for the user or not.
- *Informativeness*. The document can be considered as informative when the document has novelty and understandability.
- *Usefulness*. Whether the document is actually useful for solving the user's problem or not.

2.1.8 Retrieval experiment

Test collection

In attempt to develop a new retrieval technique or model, it is indispensable to verify empirically its effectiveness through a *retrieval experiment*. A *test collection* consisting of (a) a document set, (b) search requests and (c) lists of documents relevant to the search requests, is usually used for measuring how well the new technique or model finds relevant documents from the set in the retrieval experiment. The special search request for the retrieval experiment is often called *search topic*.

Various test collections have been constructed and employed in the history of IR research. Especially, TREC (Text REtrieval Conference) [310] in USA is well-known as a research project successfully enlarging size of document sets included in the test collection. Before TREC, most of the document sets was too small because it is necessary for creating the list of relevant documents to scan manually the entire set of documents, which prevented from adopting a large set of documents in retrieval experiments [21]. In TREC, the experiments have been organized as a collaborative project in which many research groups participate, and a list of documents relevant to a search topic has been created typically by checking manually not the entire set but a union of document sets returned by IR systems of the groups for the topic. This technique is called *pooling method*. Although the union of sets does not always include all relevant documents, it is expected that enough number of relevant documents for evaluating search effectiveness are contained in the set if each IR system sufficiently picks up relevant documents.

After TREC, the pooling method is widely adopted for creating test collections (e.g., NTCIR, CLEF and so on), and the document sets in retrieval experiments become sufficiently large. Furthermore, since TREC-type collaborative works enable to compare easily performance between the IR systems, research and development on IR have been largely prompted, and consequently, rapid enhancement of IR techniques has been brought by them since mid-1990s.

Performance measure and statistical test

(1) Precision and recall
Traditional indicators for evaluating a search output are *precision* and *recall*, which are defined as $v_p = r/n$ and $v_r = r/R$, respectively, where r is the number of relevant documents included in the search result with size n (i.e., n is the number of documents in the list) and R is the total number of relevant documents in the target database [22]. It is clear from the definitions that the two indicators measure different aspects of *search performance* (or *search effectiveness*). If many irrelevant documents are included in the search result, then precision becomes low. In contrast, recall becomes low when many relevant documents fail to appear in the search result. In most of cases, an inverse relationship between the two indicators is observed [67]. For instance, when a non-specific term in a query is replaced by a more specific one, precision often increases and recall decreases. The inverse relationship (or a kind of 'trade-off') is only an empirical regulation but gives important suggestions that it is difficult to obtain optimum search results, both the precision and recall of which are 1.0, and that it is insufficient in IR evaluation to use only either of the two indicators.

(2) F-measure
When a single-value indicator is needed for evaluation, 'harmonic mean' of the precision and recall is often computed in retrieval experiments such as $v_F = 2/(v_p^{-1} + v_r^{-1})$, which is called *F-measure*. Unlike standard 'arithmetic mean', a large value does not always have significant impact on the magnitude of harmonic mean. For example, when $v_p = 0.1$ and $v_r = 0.8$, it becomes that $v_F = 0.177$. Even if v_r increases to 0.9, the resulting value of F-measure is 0.18 and there is almost no change. In contrast, for the case that $v_p = 0.2$ and $v_r = 0.8$, v_F amounts to 0.32, which indicates that the change from 0.1 to 0.2 leads to a substantial increase of F-measure compared to

[21] The first retrieval experiment was conducted by M. Taube in 1953 [110], which was followed by the Cranfield test in UK (e.g., [68]). TREC succeeds to a part of methodology used in the Cranfield test. History on retrieval experiments and test collections was described in [290, 289].

[22] Such kind of indicators was already used in 1950s [144].

the change from 0.8 to 0.9. The F-measure has a property of 'indifference curve' which plays an important role in the theory of consumer preference in microeconomics [307], and as this example shows, a value at lower-level satisfaction has more contribution to the resulting score of F-measure.

When different weights reflecting user satisfaction should be put on precision and recall, weighted harmonic mean $1/[av_p^{-1} + (1-a)v_r^{-1}]$ is available where a is a parameter ($0 \leq a \leq 1$) indicating relative importance of precision against recall. In the case that users are assumed to prefer precision-oriented searching in an retrieval experiment, then a would be set to over 0.5.

(3) Average precision and recall-precision curve

In the case of ranked output based on a best match strategy such as vector space model, the precision is sometimes computed after extracting a set of top-ranked documents according to a cutoff level predefined as a parameter in an experimental design. As to the cutoff levels, 10, 30, 50, 100, 500 and 1000 has been conventionally used in retrieval experiments, and for example, precision of 10 top-ranked documents in a search result is often denoted by 'P@10'. Otherwise, if the total number of relevant documents R is used as the cutoff level for computing the precision, then the score is called *R-precision* [46]. The value of R is usually given a priori for each search topic in the test collection.

However, these special variations of precision are not sufficiently sensitive to changes of ranking. Suppose that there are only two relevant ones included in 10 top-ranked documents. The score of P@10 for a list in which two relevant documents are ranked 1st and 2nd respectively is identical to that in which 9th and 10th ranked documents are relevant (i.e., P@10 = 0.2 in both the cases). Therefore, most of experiments for ranked output has employed other rank-sensitive indicators such as *average precision*, which is defined by

$$v_a = \frac{1}{R}\sum_{i=1}^{n} I_r(i)v_p(i), \tag{2.37}$$

where $v_p(i)$ represents a precision score for the set from first to i-th ranked documents (i.e., 'P@i'), n denotes size of ranked output list as before (usually, $n = 1000$), and $I_r(i)$ is an indicator function such that

$$I_r(i) = \begin{cases} 0 & \text{if } i\text{-th ranked document is irrelevant} \\ 1 & \text{if } i\text{-th ranked document is relevant} \end{cases}. \tag{2.38}$$

Namely, v_a is an average of precision scores computed from the top to each relevant document respectively [46].

Let $\Delta v_a(j)$ be an amount of increment in average precision score with the change of the j-th document from irrelevant to relevant (where $j \leq n$). From Equation (2.37), it is clear that

$$\Delta v_a(j) = \frac{1}{R}\left(v_p(j) + \sum_{k=j+1}^{n} k^{-1}I_r(k)\right), \tag{2.39}$$

if R is unchanged [148]. Under an assumption that the $(j+1)$-th document is irrelevant at the time of computing $\Delta v_a(j)$, and that the j-th document is also irrelevant when $\Delta v_a(j+1)$ is considered, it becomes that $\Delta v_a(j) > \Delta v_a(j+1)$ because always $v_p(j) > v_p(j+1)$ and

$$\sum_{k=j+1}^{n} k^{-1}I_r(k) = \sum_{k=j+2}^{n} k^{-1}I_r(k), \tag{2.40}$$

according to the assumption. This result indicates rank-dependent nature of v_a. Namely, higher-ranked relevant documents contribute more to increment of the average precision score.

In retrieval experiments, the average precision score is computed for each search topic, and these scores are again averaged over all topics. This average is called *mean average precision* (MAP), which is often used for measuring effectiveness of an IR system (or model) in an experiment using a test collection. For comparing search effectiveness between two IR systems A and B, it is feasible to use the difference of their MAP scores $\bar{v}_a^A$ and $\bar{v}_a^B$,

$$\bar{v}_a^A - \bar{v}_a^B \equiv \frac{1}{H}\sum_{h=1}^{H} v_a^A[h] - \frac{1}{H}\sum_{h=1}^{H} v_a^B[h] = \frac{1}{H}\sum_{h=1}^{H}(v_a^A[h] - v_a^B[h]), \tag{2.41}$$

where $v_a^A[h]$ and $v_a^B[h]$ are average precision scores of system A and B, respectively, for h-th topic ($h = 1, \ldots, H$).

Otherwise, it is possible to compare search performance between two lists of ranked output by using *recall-precision curve*, which is a plot measuring recall on the x-axis and precision on the y-axis. For drawing the curve, first, eleven values of precision corresponding to recall levels from 0.0, 0.1, ..., 1.0 are computed for each topic. The precision value for h-th topic at recall level θ_k is denoted by $v_p[h;\theta_k]$ ($h = 1, \ldots, H$; $k = 1, \ldots, 11$). It is necessary for determining the values to apply an *interpolation* technique because there would not be always an exact value equal to a level of recall. For instance, when $R = 3$, recall scores of the search run are inevitably limited to 0.333, 0.666 and 1.0.

In this case, if relevant documents are ranked 1st, 4th and 10th respectively, then pairs of recall and precision values within a range from the top to each rank can be written as (0.333, 1.0), (0.666, 0.5) and (1.0, 0.3). An interpolation technique usually adopted in TREC [115] is to select the maximum value of precision from a set of pairs whose recall value is greater than or equal to the level θ_k, and to use it as $v_p[h;\theta_k]$. In this example, $v_p[h;\theta_k] = 1.0$ for $\theta_k = 0.0$ to 0.3, $v_p[h;\theta_k] = 0.5$ for $\theta_k = 0.4$ to 0.6, and $v_p[h;\theta_k] = 0.3$ for $\theta_k = 0.7$ to 1.0, which determine eleven precision values such as

1.0, 1.0, 1.0, 1.0, 0.5, 0.5, 0.5, 0.3, 0.3, 0.3, 0.3.

After the interpolation, a recall-precision curve can be drawn by taking an average,

$$H^{-1} \sum_{h=1}^{H} v_p[h;\theta_k],$$

as the value on the y-axis for recall level θ_k on the x-axis ($k = 1, \ldots, 11$).

If the curve of system A is positioned upper than that of system B in a recall-precision graph, then it is concluded that system A outperforms system B. A merit of using recall-precision curves is to compare search performance at different levels of recall. For example, in the case that the curve of system A is crossing that of system B at $\theta_k = 0.5$ and is upper in the range of $\theta_k = 0.0$ to 0.4, it is shown that system A outperforms at only low level of recall (i.e., upper position of its ranked output) and is not relatively effective in the range over the point of recall level such as $\theta_k = 0.5$.

(4) Statistical test

Equation (2.41) implies that a statistical test for checking difference of population mean of average precision between two systems can be executed on paired data $\{(v_a^A[1], v_a^B[1]), \ldots, (v_a^A[H], v_a^B[H])\}$ if the set of H topics is regarded as a random sample from the population. The null hypothesis is $H_0 : \mu_a^A - \mu_a^B = 0$ where μ_a^A and μ_a^B indicate population means of average precision for system A and B, respectively. For confirming whether H_0 is rejected or not from experimental data, it is necessary to compute a statistical quantity $\eta = (\bar{v}_a^A - \bar{v}_a^B)/(s_a^{AB}/\sqrt{H})$ where

$$s_a^{AB} = \sqrt{\frac{1}{H-1} \sum_{h=1}^{H} [(v_a^A[h] - v_a^B[h]) - (\bar{v}_a^A - \bar{v}_a^B)]^2}, \tag{2.42}$$

which is an estimate of standard deviation. If the sample is enough large (e.g., $H > 25$), η can be assumed to follow a normal distribution, and in the case that $|\eta| > 1.96$, the null hypothesis is rejected at 5% significance level. When the sample is small, η has to be assumed to follow a t distribution under a presupposition of normal population (i.e., differences of average precision between two systems are assumed to be normally distributed in the population). For instance, in the case that $H = 25$, if $|\eta| > 2.06$ then H_0 is rejected at 5% significance level. More specifically, this is a two-sided t test with freedom of 24.

In IR experiments, non-parametric tests are often used on the paired data [255, 316]. Suppose that there are m topics for which $v_a^A[h] > v_a^B[h]$ and $H - m$ topics for which $v_a^A[h] < v_a^B[h]$ (when ties $v_a^A[h] = v_a^B[h]$ are included, the value of H is reduced by subtracting the number of ties from it). If there is no difference of performance between system A and B (which is the null hypothesis), then it is expected that the number m follows a binomial distribution with parameter $p = 1/2$

such that

$$P(x \geq m) = \sum_{x=m}^{H} \left(\begin{array}{c} H \\ x \end{array} \right) \left(\frac{1}{2}\right)^H = 1 - \sum_{x=0}^{m-1} \left(\begin{array}{c} H \\ x \end{array} \right) \left(\frac{1}{2}\right)^H . \tag{2.43}$$

Therefore, if the probability is so small, then it would be concluded that system A outperforms B. For example, when $H = 20$ and $m = 16$, $P(x \geq 16) \simeq 0.006$, which means that the null hypothesis is rejected at 5% significance level. This is called *sign test* [23].

In the case of sign test, whether the difference $v_a^A[h] - v_a^B[h]$ is positive or negative is only considered without any attention to the magnitude of each difference. Meanwhile, *Wilcoxon signed-rank test* employs more sample information in that it is based on ranks of each absolute difference $|v_a^A[h] - v_a^B[h]|$. In this test, first, topics are ranked in descending order of $|v_a^A[h] - v_a^B[h]|$, and second, topics for which $v_a^A[h] - v_a^B[h] > 0$ are extracted. Finally, the sum of ranks is computed for the extracted topics. After the computation, whether the null hypothesis (i.e., 'no difference') is rejected or not is determined by using a value of the rank sum.

(5) Indicators based on multi-grade relevance assessment

In general, average precision is computed based on binary judgments as shown in Equation (2.38). However, it may be more natural to assign a multiple degree of relevance to each document for a given topic in the process of assessment (e.g., see [298]). For instance, four-point scale such as 'highly relevant', 'relevant', 'partially relevant', and 'irrelevant' may be more appropriate rather than dichotomous judgment determining always whether a document is relevant or not.

A conventional technique for calculating average precision from data obtained by multi-grade judgments is to reduce the multi-grade point into a dichotomous value. For example, in the CLIR task of NTCIR project [24] adopting four-scale judgments described above, a binary measure 'rigid relevance' was particularly defined by interpreting two grades of 'highly relevant' and 'relevant' as relevant and other grades as irrelevant. Also, in the case of 'relaxed relevance', the three categories, 'highly relevant', 'relevant' and 'partially relevant', are considered as relevant.

This kind of conversion allows to compute an average precision score, but some information contained in the original data is inevitably lost. An alternative strategy is to develop another indicator directly using multi-grade scores [148]. Among them, normalized (discounted) cumulated gain (*nDCG*) [137, 138] has been often used in IR experiments [25].

In order to compute the nDCG, first, a numerical value x_k has to be assigned to k-th document in the ranked list according to a result of multi-grade judgments. For example, if four-scale point was used, then it is feasible to allocate numerical values such that $x_k = 3$ for 'highly relevant', $x_k = 2$ for 'relevant', $x_k = 1$ for 'partially relevant', and $x_k = 0$ for 'irrelevant'. Next, 'discounted cumulated gain (DCG)' is computed as

$$d_c(i) = \sum_{1 \leq k < b} x_k + \sum_{b \leq k \leq i} \frac{1}{\log_b k} x_k, \tag{2.44}$$

where b is the base of the logarithm (e.g., $b = 2$). The idea behind the DCG is to reduce weights of relevant documents as their ranks decrease similarly to average precision. However, the reduction is not steep because x_k is divided by the logarithm of rank.

Also, it is feasible to consider the DCG for y_k representing a relevance degree in an ideal ranking ($y_1, \ldots, y_n$ are sorted in descending order of their degrees). For example, if there are just four relevant documents whose relevance degrees are 1, 2, 2 and 3 respectively, then these documents should be ideally ranked such that $y_1 = 3, y_2 = 2, y_3 = 2, y_4 = 1, y_5 = 0, \ldots$. The DCG for y_k is similarly computed as

$$d_I(i) = \sum_{1 \leq k < b} y_k + \sum_{b \leq k \leq i} \frac{1}{\log_b k} y_k. \tag{2.45}$$

Thus an average of nDCG up to the position n is defined as

$$v_D = d_c(n)/d_I(n). \tag{2.46}$$

[23] The sign test was used for an experiment reported in Section 5.2.

[24] http://research.nii.ac.jp/ntcir/index-en.html

[25] Other metrics based on multi-grade relevance assessment have been proposed in the IR field, which were extensively reviewed by Kishida(2005) [159]. Statistical nature of them was also discussed by this report.

(6) Other indicators used in IR experiments
In experiments using a test collection constructed by the pooling method, a ranked list may include documents not receiving any judgment because they happened to not enter into the pool for relevance assessment, and these documents are usually treated as irrelevant in the experiment. For this problem, an indicator *bpref* [44] has been proposed as a robust metric for the incompleteness of relevance assessment.

Also, mean reciprocal rank (*MRR*) is sometimes used for evaluating highly precision-oriented searches in which it is enough to find just one relevant document. For such searching, it is better that the relevant document is positioned as higher as possible in the ranked output. Therefore, an inverse of rank i of the most top-ranked relevant document (i.e., $1/i$) can be adopted as a measure. Exactly, MRR means an average of the inverse rank over all search topics.

2.2 Cross-Lingual Information Retrieval Techniques

This section reviews techniques for cross-lingual information retrieval (CLIR) [26], which would play an important role for multilingual document clustering in the process of computing similarity between a pair of documents (or a document and a cluster) written in different languages [27]. Specifically, matching strategies, translation techniques, term disambiguation in the process of translation, formal CLIR models, and MLIR methods are discussed in this section.

2.2.1 CLIR as research task

CLIR is a special type of information retrieval (IR) that enables a set of documents written in one language to be searched for queries made in another language. For instance, it would be convenient for Japanese people to be able to find documents in English by entering a search request in Japanese into the retrieval system. Especially, as the Internet has spread since the 1990s, the importance of CLIR for allowing users to access information resources written in various languages on the Internet has grown. Some search engines actually allow users to use a translation function for finding and viewing web sites in foreign languages. Another area in which CLIR becomes important is patent retrieval, by enabling searchers to locate easily related patents registered in foreign countries.

The IR research community has therefore been tackling a wide range of CLIR problems. The *Workshop of Cross-Linguistic Information Retrieval* held in August 1996 during the ACM SIGIR Conference marked a turning point for research on CLIR [28]. Since then, many retrieval experiments have been attempted, and have yielded unexpected enhancements in CLIR techniques. Intuitively, it would seem that machine translation (MT) could easily solve CLIR problems, because a CLIR task can be reduced to the execution of standard monolingual IR if search queries or the target documents are automatically translated into the other language. However, as yet there is no perfect MT system that always returns the correct results of translation, and it is suspected that such a system will be developed in the near future. Thus in the IR field, various unique methods of language processing or document ranking have been explored for improving CLIR performance [29]. Such research has also yielded deeper insights on some previously unconsidered aspects of IR.

More specifically, CLIR includes two types of retrieval:

1. bilingual information retrieval (BLIR)

[26] It is also called 'cross-language information retrieval'.

[27] This section is an extension of Kishida(2005) [149] and Kishida(2010) [153]. Literature review of this section dose not cover completely all techniques developed in the area because there are so many publications on CLIR, and particularly, studies focusing on only a particular language (e.g., Chinese, Japanese, etc.) are excluded. Other review articles [226, 234, 149] provide further information on CLIR research efforts.

[28] As widely recognized, research of developing CLIR techniques can be traced back to G. Salton's paper in 1970 [265].

[29] The rapid development of CLIR techniques since the mid-1990s is largely due to CLIR experiment workshops such as TREC [114], CLEF [39], and NTCIR [155], in which research programs of CLIR have been incorporated and many researchers worldwide have participated. Furthermore, test collections for CLIR constructed in these projects allow a wider range of researchers including those not participating in the projects to test new ideas on CLIR techniques.

2. multilingual information retrieval (MLIR)

In the case of BLIR, the documents are written in a single language different from the query language as in the above example (i.e., a Japanese-English bilingual search). Meanwhile, the target in MLIR is a heterogeneous set of documents written in two or more languages. For example, when Japanese, French and English documents are concurrently searched for a Japanese query, it is MLIR (see Figure 2.1). In general, MLIR is more complicated than BLIR as discussed later.

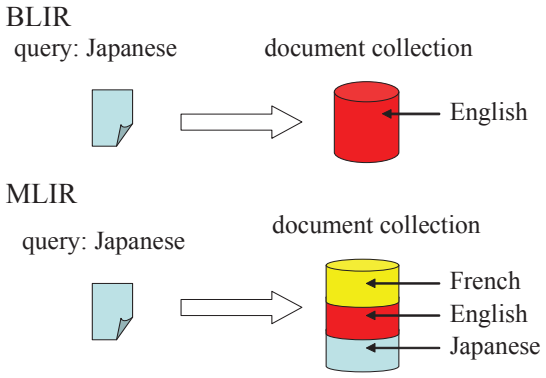


Figure 2.1: BLIR and MLIR
Source: Kishida(2010b)[153]

2.2.2 Matching strategies

Matching operation in CLIR

The most basic operation in IR is to compare the subject representation of a given search request (i.e., 'query') with that of each target document, and to measure topical similarity between them. If the degree of similarity (or relatedness) v_i between the query q and a document d_i $(i = 1, ..., N)$ were calculated by a method, then value v_i would enable to generate a search output in which documents are ranked in descending order of their similarity with the given query. As described above, this similarity is operationally defined in a retrieval model such as the vector space model, the probabilistic model and so on.

For instance, in the framework of language modeling (LM) (Section 2.1.4), v_i is computed as the probability that a set of query terms is generated from a given document d_i such as $v_i = P(\Omega_q|d_i)$ where Ω_q is a set of terms included in q. Several formulas for computing this probability have been proposed, and a simple one of which is based on Equation (2.29) as follows [126].

$$v_i = P(\Omega_q|d_i) = \prod_{t \in \Omega_q} aP(t|d_i) + (1-a)P(t), \tag{2.47}$$

where t is a term in the query and a is a mixing parameter again $(0 \leq a \leq 1)$. Actually, $P(t|d_i)$ in the formula is easily estimated as the relative frequency of term occurrence in the document d_i and $P(t)$ as the proportion of documents in which the term t appears as discussed before. Inevitably, when the query and documents are written in different languages, it is not possible to estimate $P(t|d_i)$ without relating a query term t with the corresponding term s in each document. Such kind of matching operation is important for CLIR.

There are four types of strategies for matching a query with a set of documents in the context of CLIR [225]:

- Translation
    1. Query translation
    2. Document translation

3. Interlingual technique

- No translation
  - Cognate matching

Query translation and document translation

Query translation is the most widely used matching strategy for CLIR due to its tractability, for which the retrieval system does not have to change inherent components (e.g., index files) at all in response to queries in any language if an external translation module that can convert the text of the query into the document language is incorporated. This is a remarkable advantage of query translation in practice.

However, it is difficult to resolve term ambiguity arising in the process of query translation because "queries are often short and short queries provide little context for disambiguation" [225]. Suppose that a given query is "post service". The word "post" has some different senses related to "mail", "position" and so on, and this short query does not provide sufficient information for selecting a correct translation. Many empirical analyses of transaction log data in search engines on the Internet have shown that actual queries usually consist of about two terms [204] as described above. In such a situation, the effectiveness of query translation is often limited.

Therefore, a few researchers have attempted to translate target documents into the query language (i.e., *document translation*) due to the fact that sentences included in documents tend to be more complete and to be translated correctly [226, 94, 41]. In fact, an experiment [226] showed that document translation using commercial MT software outperforms query translation in BLIR from German to English. This result suggests the potential of document translation while the short query problem remains unsolved even if documents were perfectly translated.

Furthermore, it is possible to combine the results from query translation and document translation to form a hybrid approach (see Figure 2.2). For instance, the average (or weighted average) of two document scores which were computed from query translation and document translation respectively can be used to rank documents for final output [279, 38, 140, 158]. An advantage of the hybrid approach is that it increases the possibility of correctly identifying documents having the same subject content with the query. Suppose that a term t is included in a given query and its corresponding term in the language of documents is s. If a tool for translating from the query language to the document language can not translate t into s correctly, then the system will fail to find documents containing term s by this query translation. However, when another tool for translation in the reverse direction (i.e., the document language into the query language) can identify term t from term s, matching between the query and documents including term s becomes successful.

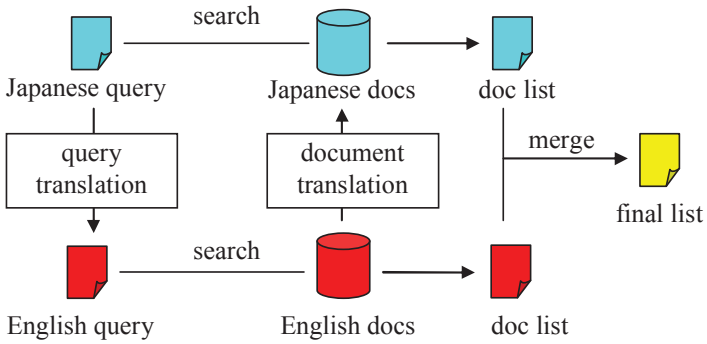


Figure 2.2: Hybrid of query and document translation
Source: Kishida(2010b)[153]

For implementing document translation or the hybrid approach, it is important to solve the

problem that document translation is a very cost-intensive task. Namely, it would take too long to translate all documents by commercial MT software. One possible solution is to employ a high-speed algorithm with low complexity at the cost of translation quality such as a 'fast document translation' algorithm [94], which is based on a statistical approach developed by IBM research group [42]. Also, simple replacement of each term in documents with its translation using a bilingual dictionary is often used as a convenient technique for document translation [30]. Of course, such dictionary-based method is also used for query translation when appropriate MT software is not available as discussed later.

Interlingual technique

In *interlingual technique*, an intermediate space of subject representation into which both the query and the documents are converted is used to compare them. There are two main categories of this technique [225]:

1. Matching within a space generated by latent semantic indexing (LSI)
2. Matching via multilingual thesauri

(1) LSI-based technique
Suppose that there is a document-aligned *parallel corpus* in which N documents are included and each document has a pair of equivalent texts in two different languages. If the two languages correspond to the query and document languages respectively, then this corpus can be used to produce an intermediate space for BLIR. An $M_1 \times N$ term-by-document frequency matrix in a language is denoted by $\mathbf{X}_1$, where M_1 is the number of distinct terms of this language contained in the corpus and each element x_{ji} of this matrix ($j = 1, ..., M_1$; $i = 1, ..., N$) is the frequency (or normalized frequency) of j-th term t_j within the text of document d_i (i.e., $x_{ji} = \mathrm{f}_{ij}$). Similarly, $\mathbf{X}_2$ is assumed to be an $M_2 \times N$ term-by-document frequency matrix for another language. Then it is feasible to constitute an $M \times N$ matrix such that

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix}, \tag{2.48}$$

where $M = M_1 + M_2$.

According to the theory of linear algebra, matrix $\mathbf{X}$ can be broken down such that $\mathbf{X} = \mathbf{U}\mathbf{Q}\mathbf{V}^T$ where $\mathbf{U}$ is an $M \times r$ orthogonal matrix, $\mathbf{Q}$ is an $r \times r$ diagonal matrix, $\mathbf{V}$ is an $N \times r$ orthogonal matrix and r is rank of $\mathbf{X}$ [31]. This decomposition is called *singular value decomposition* (SVD) (see Section 4.2.1). *Latent semantic indexing* (LSI) theory [79, 174] extracts b principal diagonal elements from $\mathbf{Q}$ ($b < r$) and interprets that they represent 'latent' meanings included in the original $\mathbf{X}$. As a result, a new indexing space $\mathbf{U}_b\mathbf{Q}_b\mathbf{V}_b^T$ is constructed for 'conceptual retrieval' where $\mathbf{U}_b$ and $\mathbf{V}_b$ are matrices in which only b columns are extracted from original ones, respectively, and $\mathbf{Q}_b$ is a diagonal matrix including only b principal elements.

Since $\mathbf{X}$ is derived from a parallel corpus, $\mathbf{U}_b\mathbf{Q}_b\mathbf{V}_b^T$ can be interpreted as a multilingual indexing space in which each meaning is represented independent of its language expressions. Therefore, if a query and a document in different languages can be projected into this indexing space, then it is possible to compare them within the space. One of the methods is to compute $\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{d}_i$ and $\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{q}$ where $\mathbf{d}_i$ and $\mathbf{q}$ are subject representations (e.g., term frequency vectors) of d_i and q, respectively [249, 175]. Note that $\mathbf{d}_i$ is an M-dimensional vector and its $(M_1 + 1)$-th to M-th elements are zero if d_i is written in the first language. Similarly, the first to M_1-th elements are zero in the vector of q in the second language. Therefore, an inner product of the two vectors $\mathbf{d}_i^T\mathbf{q}$ is always zero, but $(\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{d}_i)^T\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{q} > 0$ if d_i has semantic similarity with q in the multilingual indexing space. This means that it is feasible to use a similarity measure based on the inner product $(\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{d}_i)^T\mathbf{Q}_b^{-1}\mathbf{U}_b^T\mathbf{q}$ as v_i for ranking documents. Similar approaches were employed by some researchers (e.g, see [23, 85, 191]).

[30] This method will be adopted in MLDC system of this thesis.

[31] In diagonal matrix, all non-diagonal elements are zero. Also, when $\mathbf{U}$ is an orthogonal matrix, $\mathbf{U}^T\mathbf{U} = \mathbf{U}\mathbf{U}^T = \mathbf{I}$ where $\mathbf{I}$ is a diagonal matrix whose all diagonal elements are one. See Section 4.2.1 for details. Note that definition of $\mathbf{U}$ and $\mathbf{V}$ is slightly different here.

(2)Thesaurus-based techniques
Another type of interlingual approach is to map a term in a language into the corresponding term in another language via linguistically neutral labels of concepts registered in bilingual thesauri such as WordNet [82, 27], ULMS (Unified Medical Language System) [88] and so on. For example, it may be possible to use sense labels of 'synsets' (sets of synonymous words) provided in WordNet. The English word "train" has the sense label "train/1" implicating a line of railway cars, and it is mapped to an Italian synset including "convoglio" and "treno" in MultiWordNet[32]. The label "train/1" can be employed as an interlingua for English to Italian BLIR in the case that "train" implicating a line of railway cars is included in an English query (logically, the matching operation with this interlingua is equivalent to using directly such translations as "convoglio" and "treno" listed in the multilingual thesaurus).

It is not so easy, however, to find interlingual concepts relevant to a given query in such a multilingual thesaurus. One possibility is to search interlingual concepts (i.e., labels) for the given query by using their descriptions given in the thesaurus, and to rank interlingual concepts according to a retrieval model. For example, it may be possible to identify correctly the relevant concept "train/1" for the query "trains on a railroad" if English text in the multilingual thesaurus is searched for the query and the label "train/1" has the highest score for ranking (see [88] for details).

Cognate matching

(1)Fuzzy matching
For BLIR between two similar languages (e.g., English and French), it is possible to identify document terms equivalent to a given query term by using a fuzzy matching technique without any MT. This method is sometimes called *cognate matching*, and in the most naïve case, untranslatable terms such as proper nouns or technical terminology are left unchanged in the stage of translation. The unchanged terms are expected to match successfully a corresponding term in another language if the two languages have a close linguistic relationship.

A useful device for effectively matching cognates of two different languages is *edit distance*, which measures similarity between two character strings [75]. For example, *Levenshtein distance* is defined as the minimum number of deletions, insertions, or substitutions required to transform one string into another string [98]. Actually, since the English word "family" can be converted into the French word "famille" by inserting "l" and substituting "e" for "y", the distance between them is 2. If a pair of two terms whose distance is less than a threshold are heuristically regarded as identical terms, then term matching in BLIR can be operated with no translation. Interestingly, Buckley et al.(1998) pointed out that "English query words are treated as potentially misspelled French words", and attempted to treat English words as variations of French words according to lexicographical rules [45]. Furthermore, a kind of rule for transformation, for instance, if a Spanish term starts with "es", then "es" should be replaced by "e" for converting it into an English word, would be helpful for fuzzy matching with edit distance. Such rules may be automatically generated from statistical analysis of language resources (e.g., see [238] for details).

An alternative approach to fuzzy matching is to decompose each word in both the query and documents into n-grams (more specifically, character-based overlapping n-grams), and to perform matching operations between the two sets of n-grams [125, 211, 212]. For example, when $n = 2$ (i.e., bi-gram), "family" and "famille" are decomposed into { fa, am, mi, il, ly } and { fa, am, mi, il, ll, le }, respectively, and the similarity between them is computed. Since they have four common bi-grams, the Dice coefficient is calculated as $(2 \times 4)/(5 + 6) = 0.727$, which can be used as a metric to identify corresponding terms. It is also possible to extract and compare tri-grams ($n = 3$), quad-grams ($n = 4$) and so on. In order to enhance matching possibilities, it may be effective to transform the term beforehand according to a heuristic rule such as the Spanish-English example described above (see [304]).

(2)Machine transliteration
When two languages are very different (e.g., English and Japanese), techniques based on edit distance or n-gram inevitably will not work well. However, in such cases, phonetic transliteration

[32]http://multiwordnet.itc.it/english/home.php

from English words may be effective for cognate matching. Gey(2001) stated that "...we can often find that many words, particularly in technology areas, have been borrowed phonetically from English and are pronounced similarly, yet with phonetic customization in the borrowing language" [105].

Accordingly, it is feasible to use *machine transliteration* [161] for the operation of matching terms in very different languages. There are two methods of machine transliteration, namely, modeling of transliteration and extracting transliterations from a parallel corpus [170]. In a typical transliteration modeling, phonetic coincidence between two terms is examined. For example, the English word "America" corresponds to a Japanese word consisting of four *Katakana* characters pronounced "a", "me", "ri" and "ka", respectively (see Figure 2.3). This means that the English term "America" can be identified from the combination of four sound representations "a-me-ri-ka" in Japanese if a heuristic rule for converting "ka" to "ca" is introduced (see [97, 244] for details).

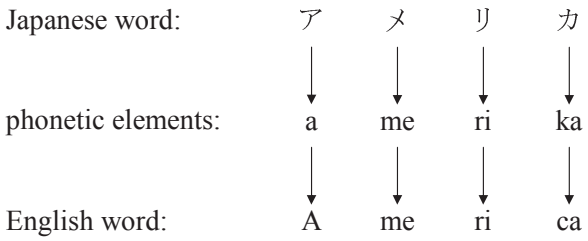


Figure 2.3: Example of machine transliteration
Source: Kishida(2010b)[153]

Also, it is possible to estimate a conditional probability $P(t|s)$ from parallel corpora as discussed later, where s is a transliteration and t indicates an original term. Such kinds of probabilities or alternative statistics indicating the relationship between t and s enable to produce a list of transliterations [170]. A combination of the rule-based method using phonetic elements and corpus-based method would be a promising strategy for correctly detecting transliterations in CLIR.

2.2.3 Translation methods

It is widely recognized that there are three main approaches to translation in CLIR:

- Machine translation
- Translation by bilingual machine-readable dictionary
- Parallel or comparative corpora-based methods

In addition, some researchers have recently attempted to use Internet resources for obtaining translation equivalents.

MT and dictionary-based methods

(1) Machine translation system
As already mentioned, intuitively, the MT system seems to be a fine tool for CLIR, and if good MT software is available, then the CLIR task would become easier. However, in query translation, the MT approach has not always shown better performance than simpler dictionary-based translation. For example, an experiment [16] indicated the dominance of dictionary-based techniques over a popular commercial MT system (of course, which method is dominant depends highly on the quality of the dictionary and MT system used for each experiment).

One of the reasons is that queries are often short and do not provide sufficient contextual information for translation. In particular, a query is often represented as only a set of terms, and it may be difficult to expect MT systems to work well with such poor representation. Also, MT systems usually try to select only one translation from many candidates that each source word may have, which may lead to removing synonyms or related terms from the set of translations,

and to missing relevant documents [222].

(2) Dictionary-based methods
Simple replacement of source terms by using a bilingual *machine-readable dictionary* (MRD) is a general approach for CLIR when no MT system with an established reputation is available. In the world, numerous languages are spoken, and there are many pairs of languages for which effective MT software has not yet been developed. Therefore, it is still important to explore CLIR methods based on bilingual MRD because such dictionaries are easier to prepare than MT software.

Most retrieval systems are still based on the so-called '*bag-of-words*' architecture, in which both the query and document texts are decomposed into a set of words (or phrases) through a process of indexing (see Section 2.1.1). Thus a query can be easily translated by replacing each query term with its translation equivalents appearing in a bilingual dictionary or a bilingual term list. Due to its convenience, the dictionary-based method is also used in document translation as described above.

Unlike standard MT systems, it is not difficult in dictionary-based methods to remain multiple translations for each source term. If they contain synonyms or effective related terms relevant to the given query, then search performance may be improved by entering them into the IR system (of course, this strategy does not always have positive effects).

However, there are some difficulties when executing this method as follows [15]:

- Specialized vocabulary not contained in the dictionary will not be translated.
- Dictionary-based translation is inherently ambiguous and may add extraneous information.
- Failure to translate multi term concepts such as phrases reduces effectiveness.

These defects are the main reasons for degradation of CLIR performance in comparison with that of monolingual retrieval. Hull & Grefenstette(1996) stated that "...we learn that translation ambiguity and missing terminology are the two primary sources of error..." [133]. Also, they reported that manual translation of multi word noun phrases improves retrieval performance. This suggests the importance of translation of multi term concepts. For example, a combination of independent translations from "hot" and "dog" would produce highly inappropriate translations of "hot dog" in many cases. Various methods have been developed for solving problems of out-of-vocabulary, term disambiguation and phrasal translation, as will be discussed later.

Practically, it is necessary for executing dictionary-based translation to compare a string of each source term with that of headwords in the dictionary. A stemming algorithm that removes a suffix from each string (see Section 2.1.6) is often employed before the matching operation in order to find successfully the corresponding term in the dictionary. For enhancing the performance of this process, it may be effective to apply *backoff translation* in which both the surface form and its stem are taken into account [227, 186]. For example, in four-stage backoff translation, four different matching operations are performed: (a) matching of the surface form of a source term to the surface form of headwords in the dictionary, (b) matching of the stem of a source term to the surface form of headwords, (c) matching of the surface form of a source term to the stems of headwords and (d) matching of the stem of a source term to the stems of headwords. Since there is no completely perfect stemmer, it is important to use a technique such as backoff translation to increase the possibility of finding the corresponding term in the dictionary.

Parallel corpora-based method

Parallel or comparable corpora are useful resources enabling to extract beneficial information for CLIR. As described already, the cross-lingual LSI approach uses this kind of corpus to construct a multidimensional indexing space. Otherwise, translation equivalents can be directly obtained from a parallel or comparable corpus by the following methods:

- Use of search results from parallel corpus
- Construction of bilingual term lists

(1) PRF-based method using parallel corpus
Suppose that English to French bilingual searches were executed by using a document-aligned parallel corpus of English and French languages. The first approach [76, 327] tries to extract French terms appearing frequently in French documents obtained from the parallel corpus by searching for the given English query (see Figure 2.4). Namely, since each French document is aligned with an English document in the corpus, it is feasible to identify French documents corresponding to English documents searched for the given English query. Naturally, such French documents are expected to include search terms relevant to the query.

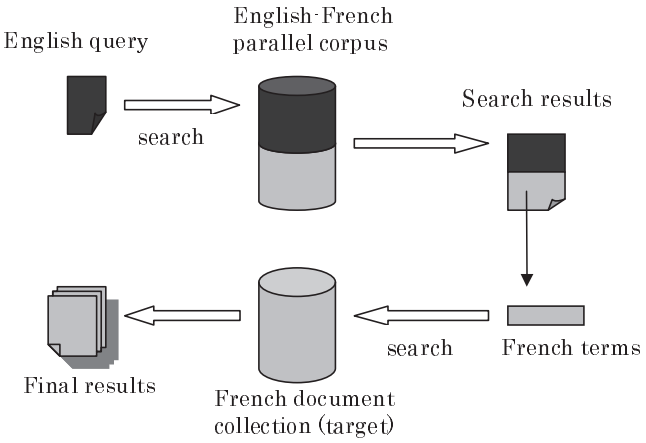


Figure 2.4: Example of BLIR using parallel corpus
Source: Kishida(2010b)[153]

This approach can be regarded as a kind of pseudo-relevance feedback (PRF), which is often used for query expansion (Section 2.1.5) in IR experiments. Its basic assumption is that top-ranked documents searched for a given query tend to be relevant and contain effective terms other than those included in the original query. According to a standard PRF technique, such effective terms can be identified from a list of terms ranked by scores computed for each term t_j as weight ω_j in Equation (2.36). By using this weight, it is possible to select some top-ranked French terms from the result of searching the parallel corpus.

(2) Estimation of association between terms
In the second approach, a bilingual term list is generated from parallel or comparable corpora based on empirical term association computed from co-occurrence statistics as shown in Table 2.2. The generated list can be used as a bilingual MRD. Suppose that t indicates a French word and s an English word. In this case, n_{ts} is the number of alignments including both t and s in a parallel corpus. From the data in Table 2.2, it is possible to calculate association τ_{ts} between t and s as the Jaccard's coefficient $\tau_{ts} = n_{ts}/(n_t + n_s - n_{ts})$ [2] or mutual information (MI) [210],

$$\tau_{ts} = \log \frac{P(t,s)}{P(t)P(s)} = \log \frac{Sn_{ts}}{n_t n_s}, \tag{2.49}$$

in which it is assumed that $P(t,s) = n_{ts}/S$, $P(t) = n_t/S$ and $P(s) = n_s/S$ where S denotes the total number of alignments (e.g., sentences) in the corpus. Also the logarithm of a likelihood ratio $-2\log\lambda$ [87] such that

$$-2\log\lambda = 2\log \frac{\zeta(p_1, n_{ts}, n_s)\zeta(p_2, n_t - n_{ts}, S - n_s)}{\zeta(p, n_{ts}, n_s)\zeta(p, n_t - n_{ts}, S - n_s)}, \tag{2.50}$$

can be used as association τ_{ts} where $\zeta(x,y,z) \equiv x^y(1-x)^{y-z}$, $p_1 = n_{ts}/n_s$, $p_2 = (n_t - n_{ts})/(S - n_s)$, and $p = n_t/S$. Furthermore, it is possible to employ other metrics such as the Dice coefficient, χ^2 statistics and so on, which are computed from a contingency table like Table 2.2.

Table 2.2: Term co-occurrence frequency in parallel corpus

	t appears	t does not appear	Total
s appears	n_{ts}	$n_s - n_{ts}$	n_s
s does not appear	$n_t - n_{ts}$	$S - n_s - n_t + n_{ts}$	$S - n_s$
Total	n_t	$S - n_t$	S

Otherwise, a method for constructing a similarity thesaurus (Section 2.1.5) can be used to compute term association between t and s if a concatenation of two aligned documents in different languages is assumed to be a single document [282, 40]. According to the standard vector space model (Section 2.1.3), it is possible to define an N-dimensional term vector whose i-th element is the weight of the term in d_i (e.g., $w_{ij} = \mathrm{f}_{ij} \times \log N/n_j$ for given t_j). If term similarity is computed as a cosine measure between the two vectors of t and s like in Equation (2.34), then query terms in the target language can be selected for each source term s according to their cosine values. Also, as another method for constructing a bilingual term list from a parallel or comparable corpus, a more complicated technique based on cognate matching and morpho-semantic analysis has been proposed (see [205]).

(3) Estimation of translation probability
Some researchers in the CLIR field (e.g., [222]) have attempted to estimate *translation probability* $P(t|s)$ from parallel corpora according to a well-known algorithm developed by a research group at IBM [42]. By executing the algorithm for a set of sentence alignments included in a parallel corpus, a bilingual term list with a set of probabilities that a term is translated into equivalents in another language is automatically generated. The algorithm includes five models, Model 1 through Model 5, Model 1 of which is the most basic and is often used for CLIR. In particular, researchers employing the language modeling approach for CLIR (see below) have used the IBM Model 1 for computing translation probabilities (e.g., [321, 167]).

The fundamental idea of Model 1 is to estimate each translation probability so that the probability defined as

$$P(\mathbf{t}|\mathbf{s}) = \frac{\varepsilon}{(l+1)^m} \prod_{j=1}^{m} \sum_{k=0}^{l} P(t_j|s_k), \tag{2.51}$$

is maximized, where $\mathbf{t}$ is a sequence (usually a sentence) of terms $t_1 t_2 \dots t_m$, $\mathbf{s}$ is the corresponding sequence of terms $s_1 s_2 \dots s_l$ and ε is a parameter (s_0 indicates an empty in $\mathbf{s}$). Namely, the translation probability $P(t_j|s_k)$ in Equation (2.51) is determined so that $P(\mathbf{t}|\mathbf{s})$, which is the probability that sequence $\mathbf{s}$ is translated into $\mathbf{t}$, takes a maximum value. Actually, each $P(t_j|s_k)$ is estimated by iterative computation as an EM algorithm [33]. The algorithm based on the IBM Model has become widely available with the release of a software package, the GIZA++ toolkit [228], incorporating it as a component.

Translation probability in the model of Equation (2.51) was extended by Kishida & Ishita(2009) [156] so as to take context terms into account such that

$$P_{\mathbf{s}}(t_j|s_k) = \sum_{h:s_h \in \mathbf{s}, h \neq k} P(t_j, s_h|s_k), \tag{2.52}$$

where s_h indicates a context term co-occurring with source term s_k in sequence $\mathbf{s}$ ($h \neq k$). Suppose that English text "post office in Japan" is translated into other language, in which the meaning of the term "post" needs to be disambiguated, and that two translations of "post" in a bilingual dictionary are assumed to be t_1 and t_2, which are related to 'mail' and 'position', respectively. For obtaining 'correct' translation t_1, the context-based translation probability, $P(t_1, \text{"office"}|\text{"post"})$, would work positively [34].

[33] Section 4.1.1 will give a general explanation on EM algorithms.

[34] The context-based translation probability $P(t_j, s_h|s_k)$ can be also estimated by an EM algorithm (see [156] for details).

Other models for statistical machine translation have also been developed. For instance, another statistical translation model by Melamed(2000) [213] was used for generating Japanese-English bilingual thesauri from bilingual corpora by [305]. In addition, estimation techniques based on EM algorithms have been proposed by some researchers (e.g., [127, 163, 55]).

(4) Availability of parallel corpus
Terms used in current affairs and people's names are often not registered in MRDs or dictionaries incorporated into MT systems. In this case, the correct translation can not be acquired. If a parallel corpus consisting of current news articles is available, then it may be possible to construct a bilingual term list including current terminology. This is a remarkable advantage of parallel corpus-based methods.

However, it would not always be possible to obtain such parallel corpus relevant to each CLIR situation, which prevents parallel corpus-based methods from being applied widely. Even if a parallel corpus in two languages corresponding to a BLIR task is available, the search performance will inevitably decrease when the subject domains covered by the parallel corpus do not match the BLIR situation [261].

One method for overcoming this problem may be to generate automatically parallel corpora from Internet resources. For instance, official web sites of organizations or institutions often have English and non-English pages with almost the same contents, from which parallel or comparable corpora could be produced (see [222, 250, 220] for details).

Also, automatic construction of document alignments from independent collections in two different languages has been explored by [297]. The fundamental operation for detecting such alignments is BLIR using an MT system or MRD. Namely, it is necessary to identify corresponding documents in one collection by searching for translations of key terms extracted from a document in another collection. If appropriate pairs of documents are detected by the BLIR, then it may be possible to find useful target terms other than translations of the key terms used as a query from the aligned documents.

Out-of-vocabulary problem

(1) Detecting unknown translations from web documents
In the case that the dictionary used for translation does not include the target entries (i.e., the *out-of-vocabulary* (OOV) situation), a special device is needed if cognate matching or machine transliteration can not be applied. Suppose that entries of a Japanese-English dictionary do not contain a particular Japanese term represented by *Kanji* characters, for which machine transliteration does not work. A promising method for resolving the OOV problem is to extract translations from Japanese web documents, in which English equivalents in parentheses are often provided for proper nouns or technical terms [57].

It may be possible to identify such English equivalents by examining frequencies of co-occurrence with the untranslated term in top-ranked documents fetched automatically from a search engine (e.g., Google) via its application programming interface (API) by searching for the untranslated term [336]. A simple heuristic rule is to select the English term co-occurring most frequently in a presupposed range of word sequence (i.e., text window) with the untranslated term. Of course, it is feasible to measure the association between two terms by using a metric such as χ^2 statistics and so on, and select translations based on its values [62].

Another approach for measuring the term association is to compare *context vectors* of two terms [62]. The context vector of term t is denoted by $\mathbf{w}_t$, j-th element of which represents the weight of the j-th term appearing in the top-ranked web documents. For example, the j-th element can be defined as $x(t, t_j) \times \log N/n_j$ where $x(t, t_j)$ means the number of times that term t_j appears with t in a text window of fixed size. Since the untranslated term and its translation possibly share common contextual terms in the top-ranked web documents, the similarity of their context vectors is expected to be relatively higher. The similarity can be measured by a cosine coefficient of two vectors $\mathbf{w}_t$ and $\mathbf{w}_s$ according to the vector space model.

(2) Combining multiple language resources
A method for reducing the possibility of the OOV problem occurring in the translation process is

to combine multiple translation resources. For instance, it is possible to merge translation results from distinct types of resources such as MRD, MT system and parallel corpus (e.g., [321]).

For merging the results, techniques of 'data fusion' or 'query combination' may be useful [139]. Suppose that there are results from two distinct MT systems. In the case of data fusion, two document scores computed from output by the two MT systems are summed for each document. Meanwhile, in query combination, before the estimation of document scores, a single query is formed by mering terms in the two translating results.

Pivot language approach

As already mentioned, it is not always possible to obtain bilingual resources for a particular pair of languages in a BLIR task. A promising technique to circumvent the problem of limited availability of linguistic resources would be the *pivot language approach*, in which an intermediate language acts as a mediator between two languages for which no bilingual resources are available. Suppose that a BLIR task between Japanese and Dutch is requested by a user. Even if any language resource between Japanese and Dutch is not available, it would be easier to find Japanese-English and English-Dutch resources since English is widely used as an international language. Thus BLIR between Japanese and Dutch can be performed via English as an intermediary without direct bilingual resources between Japanese and Dutch.

The pivot language approach would also alleviate the problem of explosive combinations of languages. For instance, if BLIR tasks have to be executed between each pair of n languages, then $O(n^2)$ resources are needed [35]. However, the pivot language approach enables us to handle the complex job with only $O(n)$ resources [105].

A basic pivot language approach is *transitive translation* of a query using two bilingual dictionaries [17]. In the case of searches from Japanese to Dutch via English, if Japanese-English and English-Dutch MRDs are available, then CLIR can be performed by replacing Japanese query terms with the corresponding English equivalents and successively substituting the English equivalents with the Dutch equivalents. Of course, when Japanese-English and English-Dutch MT systems can be used, a similar transitive translation is also feasible.

In dictionary-based transitive translation, translation ambiguity becomes a more serious problem. Suppose that a Japanese source query consists of three terms, and every term has three English equivalents. If every English equivalent has also three Dutch equivalents, simple replacements will produce 27 ($= 3^3$) search terms in total from only three source terms, and the final set of search terms will inevitably contain some irrelevant translations [157] (see Figure 2.5). Therefore, it is important to apply a translation disambiguation technique (discussed later) to the set of intermediary or target terms obtained from the bilingual dictionaries. Actually, an experiment [159] reported that translation disambiguation for intermediary terms did not have a remarkable effect in German-Italian bilingual searches via English and that it was enough to disambiguate only final Italian terms.

Retrieval experiments on the dictionary-based query translation via a pivot language have been conducted for various combinations of languages such as English → French → German [93], French → English → German [106], German → English → Italian [128], Japanese → English → Chinese [188], Chinese → English → Japanese [57], and so on (these are only a portion of many experiments). In particular, Franz et al.(1999) proposed some interesting techniques for searching German documents with English queries [93] as follows (the intermediary is French):

- Convolution of translation probability. Estimating translation probability from an English term s to a German term t through French terms f such that $P(t|s) = \sum_f P(t|f)P(f|s)$.
- Automatic query generation from the intermediate language corpus. Generating French queries automatically by simply merging all non-stopwords in the top-ranked French documents searched by the English-French BLIR system, and entering the French query into the French-German BLIR system.

[35] Exactly, there are $n(n-1)/2$ pairs for n languages. $O(\cdot)$ is big-O notation. Roughly speaking, if $f(x) = O(g(x))$, then there exits numbers $K > 0$ and $\delta > 0$ such that $|x| < \delta$ and $|f(x)|/|g(x)| \leq K$ (see p.72 in [63]).

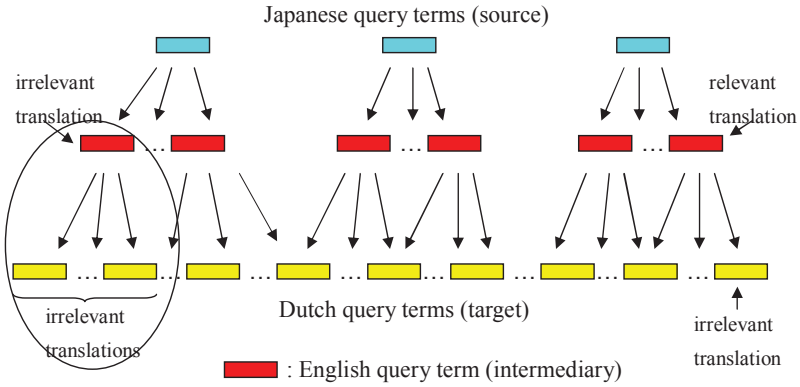


Figure 2.5: Example of dictionary-based query translation via pivot language
Source: Kishida(2010b)[153]

Translation quality

The quality or correctness of the translation largely affects CLIR performance. For example, experiments by [210, 320] have shown that the size or lexical coverage of MRD has an influence on the effectiveness of CLIR. Namely, as the number of entry terms in the dictionary is reduced and as its translation quality becomes lower, search performance gradually deteriorates.

According to Kishida(2008) [151], the CLIR performance can be represented as a regression model with two independent variables, translation quality z_1 and 'ease of searching for query' z_2,

$$y = a + b_1 z_1 + b_2 z_2, \tag{2.53}$$

where y denotes CLIR performance, and a, b_1 and b_2 are parameters. The 'ease of searching for query' is an inherent nature of a given query, which is independent of the translation process. For instance, if the query does not contain any specific concept that explicitly designates its content, then it will be difficult to find relevant documents even if the translation is perfectly correct. Therefore, it is necessary to take the 'ease of searching for query' into consideration in the regression model [151].

In retrieval experiments in the laboratory, the 'ease of searching for query' z_2 can be measured by evaluation indicators (e.g., average precision) representing the performance of monolingual searches for which correct translations of queries are given by human experts. Similarly, it is possible to gage translation quality z_1 by a metric for assessing automatically translation results such as well-known BLEU [230] based on the correct translations. In particular, a metric called WAMU (weighted average for matching uni-grams) that is specifically designed to evaluate translations in CLIR situations was developed by Kishida(2008) [151]. A simpler formula of WAMU is

$$WAMU = \left(\sum_{j=1}^{m} \omega_j \right)^{-1} \sum_{j=1}^{m} \omega_j \frac{\min(c_j, \tilde{c}_j)}{c_j}, \tag{2.54}$$

where c_j is the total number of the j-th uni-gram (i.e., a word) in the translation, $\tilde{c}_j$ is the total number of the j-th uni-gram in an answer given by a human expert (i.e., a correct translation), ω_j is the weight of the j-th uni-gram, and m is the number of uni-grams in the translation. The weight ω_j is calculated such that $\omega_j = \log(N/n_j)$ (in CLIR situations, n_j can be obtained from the target document collection).

Suppose that two translations are obtained with an answer as follows:

- Translation 1: Database management system *issue*
- Translation 2: Database *administration* system problem
- Answer: Database management system problem

Table 2.3: Sample data for calculating WAMU

j	Terms t_j	$\min(c_j, \tilde{c}_j)$	n_j	$\log(N/n_j)$
	Translation 1			
1	database	1	5	5.30
2	management	1	50	3.00
3	system	1	50	3.00
4	issue	0	100	2.30
	Total	-	-	13.59
	Translation 2			
1	database	1	5	5.30
2	administration	0	50	3.00
3	system	1	50	3.00
4	problem	1	100	2.30
	Total	-	-	13.59

In either of the translations, only one term is different from the corresponding term in the answer (i.e., "issue" in Translation 1 and "administration" in Translation 2). If these translations are to be used as a query, then Translation 1 is clearly better because it represents "database management system" correctly. When the numbers of documents including each term are as shown in Table 2.3 and $N = 1000$, the WAMU scores for Translations 1 and 2 are computed as follows:

- Translation 1: $(5.30 + 3.00 + 3.00)/13.59 = 0.83$
- Translation 2: $(5.30 + 3.00 + 2.30)/13.59 = 0.78$

As this example indicates, WAMU differentiates adequacy of translation between specific and non-specific terms. Namely, mistranslation of a specific term reduces its WAMU score more largely than non-specific terms due to idf factor $\log(N/n_j)$.

An experiment in Kishida(2008) [151] showed that the regression model containing two independent variables, 'ease of searching for query' and translation quality, can explain approximately 60% of the variation in CLIR performance. In this result, translation quality has a statistically significant effect, which means that translation quality is crucial for enhancing CLIR effectiveness, as would be expected.

2.2.4 Term disambiguation techniques

Translation ambiguity

For improving translation quality, it is indispensable to select a correct translation from a set of candidates by disambiguating the sense of a given source term. In general, *word sense disambiguation* (WSD) is a key element in various applications such as machine translation, information retrieval and hypertext navigation systems, content and thematic analysis, grammatical analysis, speech processing, and so on [134]. For CLIR, it is often necessary to disambiguate translations enumerated under each headword in a bilingual MRD or a term list generated from a parallel corpus.

If all translations listed in bilingual resources are straightforwardly adopted as search terms, then extraneous or superfluous terms irrelevant to the original query usually diminish the effectiveness of the search. Thus it is desirable that only relevant terms will be automatically or semi-automatically selected from a set of translation candidates.

A simple method is to take only translations corresponding to the first sense listed in the dictionary. Alternatively, it may be possible to investigate frequency of each translation within a corpus and use only the most frequent translation. However, such heuristic strategies would be insufficient to resolve the ambiguity of words that are highly homonymous.

Several more sophisticated methods have been explored in the field of CLIR as follows:

1. Use of part-of-speech (POS) tags

2. Use of parallel corpora

3. Use of co-occurrence statistics in the target corpus

4. Use of query expansion technique

It should be noted that the term 'corpus-based disambiguation' is often used in literature for collectively referring to techniques 2 through 4.

As another solution, the 'structured query model' has also been investigated for improving search performance in cases where multiple translations are obtained from a bilingual dictionary. This model will be discussed in another section.

Use of part-of-speech tags

The basic idea of using part-of-speech (POS) tags for translation disambiguation is to select only translations having the same POS tag as that of the source query term [75, 78, 16]. For instance, in the case of applying it to the BLIR task from English to Spanish, the translation is selected as a search term only if the POS tag of a Spanish equivalent listed in an English-Spanish dictionary coincides with that of the English query term. This technique requires POS tagging tools for both languages.

Parallel corpus-based methods

It is possible to determine the best translations among the candidates according to the result of searching a parallel corpus for the original query [75, 77, 36]. Specifically, a typical procedure is as follows (see also Figure 2.6):

1. Identify a set of translations for each term in a given source query by using a bilingual MRD.

2. Search a part of the parallel corpus written in the target language for each translation respectively, and save each set of documents in the target language.

3. Search the other part of the parallel corpus written in the source language for the source query.

4. Select a translation whose set of documents is the closest to the set of documents searched by the source query (e.g., choose a translation whose documents are mostly included in the set of search results by the source term).

This procedure is repeated for each query term until a final set of the best translations is obtained.

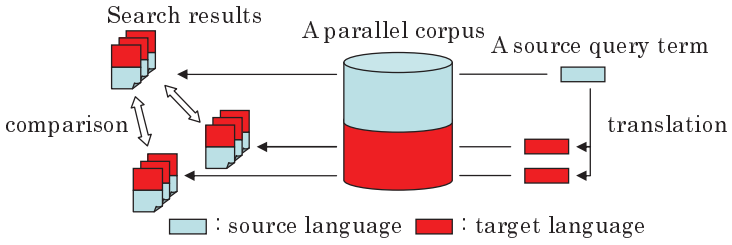


Figure 2.6: Outline of parallel corpus-based disambiguation
Source: Kishida(2010b)[153]

Disambiguation based on term co-occurrence statistics

A basic assumption underlying disambiguation techniques based on term co-occurrence is that "the correct translations of query terms should co-occur in target language documents and incorrect translations should tend not to co-occur" [16]. For example, in the case of a French to English search, when the source query includes a concept of "database management system", the combination of translations, "database", "management" and "system" is reasonably expected to appear more frequently in the target document collection than that of "database", "administration" and "system". Thus if statistics on co-occurrence frequencies are appropriately employed, then "management" would be able to be identified as a correct translation in the context [36].

This kind of methods based on term co-occurrences does not need any extra corpus because the statistics can be obtained from the target documents for searching. This is a remarkable merit in practice.

(1) Best pairs selection
When a given query includes many terms, the frequency of their combination may become very low or zero. Thus the query is usually decomposed into a set of term pairs, and co-occurrence frequencies of the corresponding two translations are used for determining final query terms $\{\tilde{t}_1, ..., \tilde{t}_m\}$ in the target language. For instance, it is feasible to select them such that

$$\tilde{t}_j = \arg\max_{t \in T_j} \text{sim}(t, t'), \quad t' \in \bigcup_{k \neq j} T_k;\ j = 1, ..., m, \tag{2.55}$$

where T_j indicates a set of translations in the target language for j-th source term s_j $(j = 1, ..., m)$ and $\text{sim}(t, t')$ is a measure of term association or similarity between the t and t' computed from term co-occurrences [37]. Equation (2.55) simply means that a pair of translations with the highest similarity is repeatedly chosen from those excluding pairs that have already been selected [26]. Namely, if a pair with the highest similarity includes a translation of the source term for which a translation was previously determined, then this translation is ignored. As a variation of this method, it may be possible to consider only similarities with the translations selected in the previous step [136].

The term similarity $\text{sim}(t, t')$ can be measured as MI, the Dice coefficient and so on, which are computed from term occurrence and co-occurrence statistics in the target document collection. For instance, if MI in Equation (2.49) is employed, then its probabilities are operationally defined such that $P(t, s) = c(t, s)/S$, $P(t) = c(t)/S$ and $P(s) = c(t)/S$ where $c(t)$ and $c(s)$ are the numbers of sentences including terms t and s, respectively, $c(t, s)$ is the number of sentences containing both terms, and S is the total number of sentences included in the target collection. While the number of observations is normalized by the size of the corpus according to [64] in these formulas, other definitions can be also adopted (e.g., see [101]). The term co-occurrence statistics required for computing the similarity can be compiled in a process of constructing index files for searching the target collection, or obtained from a search engine on the Internet via API [201].

Fortunately, the computational complexity of selecting the best pairs based on Equation (2.49) is not so high. If the similarity measure used in this process is symmetric such as MI, then the number of pairs to be measured for the selection, M_p, amounts to

$$M_p = \sum_{k=1}^{m-1} \sum_{h=k+1}^{m} |T_k| \times |T_h| \tag{2.56}$$

(see Kishida(2007) [150]). It should be noted that M_p does not exceed $b^2 m(m-1)/2$ where $m(m-1)/2$ corresponds to the number of all combinations of two source terms and b indicates the maximum number of translations for a source term (i.e., $b = \max_{j=1,...,m} |T_j|$). For example, if there are five source query terms and every set of translations includes five words respectively,

[36] Moreover, it is possible to incorporate term co-occurrence statistics into the process of estimating translation probabilities (see [194] for details).

[37] In the equation, 'arg max' means 'argument of the maximum'. For instance, $\arg\max_x f(x)$ indicates a particular value of variable x providing the maximum of function f.

it becomes that $M_p = (5 \times 5) \times (5 \times 4/2) = 250$.

(2) Best sequence selection
However, the best pair selection algorithm may yield erroneous translations because it checks only a local relationship between just two translations at each step for selecting a pair. Suppose that, in the above example, the pair of "administration" and "system" has the highest degree of similarity in the target document collection even though the collection contains some documents on "database management system". In this case, the correct translation, "management", is never obtained by the algorithm based on Equation (2.55). Since this algorithm looks at only a very limited range (i.e., a span of just two terms) in the query, a few pairs having strong relationships out of the context of the query may have impact excessively on the result.

A straightforward solution to this problem is to focus on all relationships between the query terms [281]. A sequence of translations is denoted by $\tilde{\tau} = \{t_1, ..., t_j, ..., t_m\}$ where $t_j \in T_j$ in this case. Namely, $\tilde{\tau}$ is constituted by taking arbitrarily a term from each translation set, respectively (see Figure 2.7). If the sum of degrees of similarity between all translations included in such a sequence,

$$U(\tilde{\tau}) = \sum_{k=1}^{m-1} \sum_{h=k+1}^{m} \mathrm{sim}(t_k, t_h), \quad t_k, t_h \in \tilde{\tau}, \tag{2.57}$$

can be computed, then the sequence with the highest value of the sum should be selected as the final set of translations (similar strategies were adopted by [201, 245]). Since Equation (2.57) contains all relationships between possible pairs of translations in each sequence, it is feasible to avoid errors caused by the locality of range in Equation (2.55). For example, the possibility that "management" is correctly selected remains even if the pair of "administration" and "system" has the highest similarity.

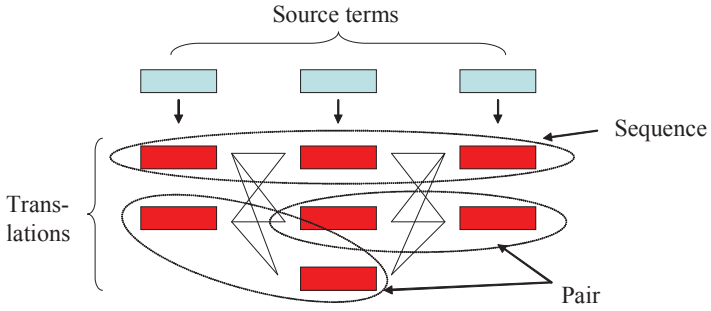


Figure 2.7: Sequences and pairs of translations
Source: Kishida(2010b)[153]

However, computational complexity of processing the algorithm based on Equation (2.57) is very high. The total number of sequences to be processed is given by

$$M_\tau = |T_1| \times ... \times |T_m| = \prod_{j=1}^{m} |T_j|, \tag{2.58}$$

and the upper limit of M_τ amounts to b^m. In addition, each sequence includes $m(m-1)/2$ pairs as expressed in Equation (2.57), and therefore, in total, it is necessary to deal with $b^m m(m-1)/2$ pairs at most in the process of disambiguation (note that a single pair is repeatedly counted). If there are five source query terms and every set of this translation includes five words respectively, then the number of pairs amounts to 31,250 ($= 5^5 \times 5 \times 4/2$). As this example shows, it takes much longer to select translations when using Equation (2.57).

(3)Approximation for best sequence selection
One way to reduce the computational difficulty of the best sequence method is to apply the

maximum value of similarity between a given translation and those of another source query term,

$$C(t, T_k) = \max_{t' \in T_k} \mathrm{sim}(t, t'), \quad t \notin T_k, \tag{2.59}$$

which was employed in some experiments (e.g., [1, 101]). $C(t, T_k)$ is often called 'cohesion'. In order to solve ambiguity of translations for each term t by using this quantity, the sum of $C(t, T_k)$ has to be computed over all sets T_k except the set that includes the term t itself, and then to select the term with the highest score from each set of translations respectively [1, 101], which means that

$$\tilde{t}_j = \arg\max_{t \in T_j} \sum_{k \neq j} C(t, T_k), \quad j = 1, ..., m. \tag{2.60}$$

The number of translation pairs to be processed in this algorithm for determining the translation of the j-th source term is $|T_j| \times \sum_{k \neq j} |T_k|$, and therefore, the total number amounts to

$$M_p = \sum_{j=1}^{m} \left(|T_j| \times \sum_{k \neq j} |T_k| \right), \tag{2.61}$$

which does not exceed $m \times b \times (m-1)b = b^2 m(m-1)$. If there are five source query terms and every set of translations includes five words respectively, then it becomes that $M_p = 5 \times 5 \times 4 \times 5 = 500$.

Although its degree of computational complexity is almost the same as that of the method in Equation (2.55), the algorithm using $C(t, T_k)$ avoids making local judgments in the selection of translations to some degree unlike Equation (2.55). Namely, a translation of a given source term s_j is chosen by using information on relationships with all other source terms through the $C(t, T_k)$ in Equation (2.59), not with only a single source term. In the above example, even if the similarity score between "administration" and "system" is the highest one, it may be possible that "management" is correctly selected as a translation. Actually, the final sum of $C(t, T_k)$ for "management" may become greater than that for "administration" due to the contribution of a high degree of similarity between "management" and "database". Search performance of the disambiguation techniques will be discussed in Section 5.2 based on an experimental result provided by Kishida(2007) [150].

PRF-based techniques for disambiguation

PRF techniques discussed in Section 2.1.5 can be applied to translation disambiguation [15, 16]. In CLIR situations, two kinds of PRF are feasible:

- Pre-translation feedback
- Post-translation feedback

Suppose that there is a corpus in the source language which is independent of the target document collection. First, this corpus is searched for a given source query, and resulting documents are analyzed prior to translation for CLIR in order to add a set of new terms to the source query (*pre-translation feedback*). The new terms can be selected based on term weights ω_j in Equation (2.36). Second, after translation, standard PRF can be applied by using the target document collection (*post-translation feedback*). Inevitably, for executing the pre-translation feedback, an extra corpus in the source language is needed unlike the post-translation feedback working on the target collection.

The pre-translation feedback may improve the precision (see [15]), because PRF is basically executed using the entire query, not each source term respectively. Namely, synonyms or related terms corresponding to the correct meaning of each source term within the context of the query are expected to be automatically added through the PRF process. An experiment [210] has suggested that the pre-translation query expansion is useful when lexical coverage of translation resources is poor. In contrast, the post-translation feedback is just standard PRF, and therefore, the recall would increase by applying it as many IR experiments have shown.

It is also possible to determine explicitly a final translation $\tilde{t}_j$ for each source term ($j = 1, ..., m$) based on frequencies of term occurrence in output from the initial search of post-translation

feedback according to Kishida & Kando(2004)[157]. In the first stage of this process, the target document collection is searched for a set of all translation candidates, and the number of documents including each translation candidate in the list of top-ranked documents by this search is counted. Finally,

$$\tilde{t}_j = \arg\max_{t \in T_j} r_t, \quad j = 1, ..., m, \tag{2.62}$$

is selected as a final translation for the j-th source term where r_t indicates the number of documents including a translation candidate t within the top-ranked documents.

As Yamabana et al.(1998) [325] pointed out, unexpected false combinations of translations may be generated by disambiguation techniques based on term co-occurrences because it is possible that two translations having no relation within the context of a given source query tend to co-occur frequently in the whole document collection. Namely, suggestions from macro-statistics compiled by using the entire collection are not always valid in the sense implied by a particular query. A similar problem occurs when applying query expansion techniques to general IR situations. It is widely known that query expansion techniques using similarity thesauri generated through term co-occurrence statistics (see Section 2.1.5) cannot achieve better performance than PRF in which new search terms are locally identified from a restricted set of top-ranked documents searched for the given query. Similarly, in the context of translation disambiguation for CLIR, local analysis of the target document collection such as Equation (2.62) may yield better results than global analysis.

In addition, it should be noted that the technique based on Equation (2.62) is easier to implement in IR systems having a standard PRF function. This is a practical advantage of this technique.

Disambiguation for phrasal translation

As Ballesteros & Croft(1997) [15] pointed out that "...failure to translate multi term concepts as phrases reduces effectiveness", *phrasal translation* is certainly significant for CLIR (see also Section 2.2.3). A basic technique is to search a bilingual dictionary or a term list including phrases or compound words as its headwords. Namely, they can be automatically identified in the source query by simple matching operations against headwords of such a language resource. Also, if a POS tagger is available in this process, then a word combination of 'noun-noun' or 'adjective-noun' would be reasonably assumed to be a compound word or multi-word (see also Section 2.1.6).

Inevitably, the coverage of lexical resources to be used will not always be sufficient. Namely, all phrases or compound words are not always found in the resources as headwords. When an untranslatable phrase is included in a given source query, there is no way other than to attempt word-by-word translation, which may cause a term ambiguity problem [15, 16]. Therefore, one of the disambiguation techniques discussed above is required in the word-by-word translation process.

However, it should be noted that there are two kinds of compound words as follows [237]:

- Compositional compounds. The meaning can be derived from meanings of the component words (e.g., "database management system").
- Non-compositional compounds. The meaning cannot be derived from meanings of the component words (e.g., "hot dog").

In many cases, disambiguation methods based on term co-occurrences may have limitation for detecting correctly translations of non-compositional compounds. It is thus indispensable to augment the coverage of bilingual dictionaries in order to enhance the quality of phrasal translations. This could be done by some techniques for detecting translations of OOV from the web (see the above section), for extracting phrasal representations from parallel or comparable corpora (see [197]), or for disambiguating directly noun phrases not at the level of distinct words (see [100]).

Other disambiguation techniques

In a technique called *bi-directional translation*, backward translations in which translation results are automatically re-translated into the original language are used for ranking translation candidates [36]. For instance, when French terms are translated into English ones, first a set of English

equivalents for each French term is extracted from a French-English bilingual dictionary. Next, by using an English-French dictionary, each English equivalent is reversely translated into a set of French terms. Basically, if the set includes the original source term, then the English translation equivalent is chosen as a preferred translation.

When a pivot language approach to translation is used, *lexical triangulation* can be applied to translation disambiguation [108, 184]. In this technique, two pivot languages are used independently, and an attempt is made to remove erroneous translations by taking only translations in common obtained from both ways of transitive translation using the two pivot languages respectively.

Also, a disambiguation technique via dynamic clustering of search results has been explored by [181]. In the first stage of this method, all translation candidates are used as the search query without any disambiguation like the PRF-based method, and in the next stage, top-ranked documents searched for the original query are divided into some clusters by a clustering algorithm. Finally, the degree of similarity between the query and a cluster to which each document belongs is reflected in its document score for ranking, and documents are re-ranked according to the new scores. If each cluster correctly corresponds to one of the multiple senses and appropriate clusters for the given query can be identified, then search performance may be enhanced by the re-ranking.

2.2.5 Formal models for CLIR

If representations in two different languages of a given query and documents become comparable after the translation process, then the BLIR problem reduces to that of standard monolingual IR. Namely, after the translation is completed, the remaining task is how to compute document scores for ranked output, in which a retrieval model such as Equation (2.12), (2.28) or (2.47) is applied in a standard manner of monolingual IR. In contrast, some researchers have tried to incorporate directly the translation process into a retrieval model.

Language modeling for CLIR

It is relatively easy to incorporate translation probabilities $P(t|s)$ into Equation (2.47) of language modeling (LM). Figure 2.8 shows the basic idea of the incorporation, in which it is assumed that

$$P(t|d_i) = \sum_k P(t|s_k)P(s_k|d_i), \tag{2.63}$$

where s_k indicates a term included in document d_i. Note that t is a query term in a different language from that of documents in this case. Equation (2.63) means that $P(t|d_i)$, which is the probability that query term t is generated from document d_i, is decomposed into term occurrence probabilities $P(s_k|d_i)$ and translation probabilities $P(t|s_k)$, and that its value is computed by summing the products of them for all s_k. By directly substituting Equation (2.63) into Equation (2.47),

$$P(\Omega_q|d_i) = \prod_{t\in\Omega_q} a\sum_k P(t|s_k)P(s_k|d_i) + (1-a)P(t), \tag{2.64}$$

is finally obtained according to [215, 321, 320].

In Equation (2.47), for preventing $P(\Omega_q|d_i)$ from automatically becoming zero when $P(t|d_i) = 0$ for term t, a general probability $P(t)$ is added to $P(t|d_i)$ with a mixing parameter a (i.e., Jelinek-Mercer smoothing) where the term $aP(t|d_i) + (1-a)P(t)$ can be considered to be the 'true probability' that t appears in d_i. If translation probabilities are applied to the true probability that term s_k appears in d_i, then another formula,

$$P(\Omega_q|d_i) = \prod_{t\in\Omega_q} \sum_k P(t|s_k)[aP(s_k|d_i) + (1-a)P(s_k)], \tag{2.65}$$

can be derived for language modeling in BLIR [128]. Whereas $P(t)$ in Equation (2.64) needs to be estimated using an extra corpus in the query language (e.g., $\hat{P}(t) = n_t/N$ where n_t means the number of documents containing term t), an estimation of $P(s_k)$ in Equation (2.65) can be obtained from the target document collection itself.

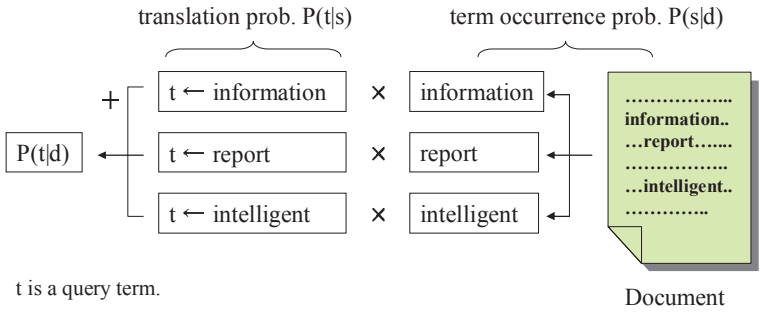


Figure 2.8: Language modeling with translation probabilities
Source: Kishida(2010b)[153]

As discussed above, translation probability $P(t|s_k)$ can be estimated by the IBM model if an appropriate parallel corpus is available. Fortunately, the GIZA++ toolkit including a module for the IBM model has been developed.

If no parallel corpus is available, then other methods have to be employed. A simple one is to count the number of translations for each source term in a bilingual dictionary. For example, if a source term s has m_s translations $t_1, ..., t_{m_s}$, then the translation probabilities can be assumed such that $P(t_j|s) = 1/m_s$ $(j = 1, ..., m_s)$ uniformly [321]. As a more sophisticated method, an EM algorithm computing iteratively the translation probabilities based on term co-occurrence statistics has been proposed by [217] (the initial values of translation probabilities are estimated as the above $P(t_j|s) = 1/m_s$ from a bilingual dictionary). Since term co-occurrence statistics can be compiled from the target document collection, this algorithm allows to estimate translation probabilities without any parallel corpus.

Relevance model

Relevance model (RM) developed by [179] can be theoretically applied to CLIR situations [178, 176] since essentially the RM is a variation of LM. In RM, the document score is often calculated as relative entropy $E(\mathcal{R} \parallel d_i)$ between $P(t|d_i)$ in Equation (2.47) and $P(t|\mathcal{R})$, which is the average probability that t is included in a relevant document, such that

$$E(\mathcal{R} \parallel d_i) = \sum_{t \in \Omega_D} P(t|\mathcal{R}) \log \frac{P(t|\mathcal{R})}{P(t|d_i)}, \tag{2.66}$$

where Ω_D indicates a set of all terms appearing in the database D. If a set of relevant documents is unknown, then $P(t|\mathcal{R})$ can be approximately estimated by $P(t|\Omega_q) = P(t, \Omega_q)/P(\Omega_q)$, and it is feasible to write that

$$P(t, \Omega_q) = \sum_{i:d_i \in D} P(d_i) P(t|d_i) \prod_{t' \in \Omega_q} P(t'|d_i), \tag{2.67}$$

under an assumption [179] [38]. Also, it becomes

$$P(\Omega_q) = \sum_{t \in \Omega_D} P(t, \Omega_q). \tag{2.68}$$

In order to apply RM to CLIR, it is necessary to distinguish terms in documents from those in queries by denoting a document term by s_k in Equations (2.66) and (2.67) (i.e., $s_k \in \Omega_D$). Actually, Equation (2.67) is re-written as

$$P(s_k, \Omega_q) = \sum_{i:d_i \in D} P(d_i) P(s_k|d_i) \prod_{t \in \Omega_q} P(t|d_i), \tag{2.69}$$

[38] First, $P(t, \Omega_q) = \sum_{i:d_i \in D} P(d_i) P(t, \Omega_q|d_i)$. If it is assumed that all terms are sampled independently and identically from d_i, then $P(t, \Omega_q|d_i) = P(t|d_i) \prod_{t' \in \Omega_q} P(t'|d_i)$. By substituting it into the first equation, Equation (2.67) is obtained.

in which $\prod P(t|d_i)$ can be estimated by using Equation (2.64) and usually $P(d_i) = 1/N$.

While only query terms appearing in the document contribute to computation of the document score in standard IR models, RM takes all terms in the database D into consideration as suggested by Equation (2.66). This means that much more information on term occurrences is used in RM than other IR models, which may enhance search performance. However, more computational time is required for calculating document scores based on RM.

Hidden Markov model for CLIR

Suppose that $\mathbf{s}$ is a sequence of terms included in a source query and $\mathbf{t}$ is its translation, and that the number of source terms is equal to that of translations (e.g., $m = l$ in Equation (2.51)). From the definition of conditional probability, $P(\mathbf{t}, \mathbf{s})$ can be written as

$$P(\mathbf{t}, \mathbf{s}) = P(t_1, ..., t_m, s_1, ..., s_m) = P(s_1, ..., s_m|t_1, ..., t_m)P(t_1, ..., t_m), \tag{2.70}$$

where $P(t_1, ..., t_m) = P(t_m|t_{m-1}, ..., t_1) \times ... \times P(t_3|t_2, t_1) \times P(t_2|t_1) \times P(t_1)$, and if the translation process is interpreted as a Markov model in which occurrence probability of t_h is dependent on only previous term t_{h-1}, then it becomes that

$$P(t_1, ..., t_m) = P(t_m|t_{m-1})...P(t_3|t_2)P(t_2|t_1)P(t_1). \tag{2.71}$$

Similarly, $P(s_1, ..., s_m|t_1, ..., t_m) = P(s_m|t_m) \times ... \times P(s_2|t_2) \times P(s_1|t_1)$, and therefore,

$$P(\mathbf{t}, \mathbf{s}) = P(t_1)P(s_1|t_1) \prod_{j=2}^{m} P(s_j|t_j)P(t_j|t_{j-1}). \tag{2.72}$$

This is a query translation model based on hidden Markov model (HMM) proposed by [91, 24], in which 'reverse' translation probability $P(s_j|t_j)$ and transitive probability $P(t_j|t_{j-1})$ of translations (target terms) are included. The transitive probability in this model may work as a device for translation disambiguation. For example,

$P(\text{“management”}|\text{“database”}) \times P(s_2|\text{“management”})$

is expected to be greater than

$P(\text{“administration”}|\text{“database”}) \times P(s_2|\text{“administration”})$

where s_2 is the second term of a query representing “database management system” in a language other than English.

Structured query model

INQUERY system developed by a research group at Massachusetts University [47, 306] provides an important function for CLIR. In principle, the INQUERY is a retrieval model based on a probabilistic Bayesian network, in which a 'belief score' $B(q|d_i)$ measuring the degree to which document d_i is relevant to a given query q is estimated. Although some variations can be derived dependently on assumptions to be selected, $B(q|d_i)$ is basically computed from $B(t_j|d_i)$ of each search term included in q such that

$$B(t_j|d_i) = 0.4 + 0.6 \left\{ \frac{\mathrm{f}_{ij}}{\mathrm{f}_{ij} + 0.5 + 1.5l_i/\bar{l}} \times \frac{\log[(N + 0.5)/n_j]}{\log(N + 1)} \right\}, \tag{2.73}$$

where l_i is the length of d_i, $\bar{l}$ is the average of l_i in the database and f_{ij} is the frequency of t_j in document d_i, as before. For example, when '#sum( )' operator is used such as #sum$(t_1, t_2, ..., t_m)$, $B(q|d_i)$ is calculated as $\sum_{j=1}^{m} B(t_j|d_i)$.

The INQUERY system has several operators like #sum( ), which clearly distinguish this system from other retrieval models. Among the operators, the #syn( ) operator for dealing with synonyms is often used in CLIR tasks. Suppose that there are three source terms s_1, s_2, s_3, and translations of these terms are obtained from a bilingual dictionary such that $T_1 = \{t_{11}, t_{12}, t_{13}\}$, $T_2 = \{t_{21}, t_{22}\}$ and $T_3 = \{t_{31}, t_{32}\}$. It is feasible to enter a structured query,

#sum (#syn (t_{11} t_{12} t_{13}) #syn (t_{21} t_{22}) #syn (t_{31} t_{32})),

into the INQUERY system without any translation disambiguation [236]. This is often called

Pirkola's method, in which the belief score of a set of translations T_k for the source term s_k is computed based on the #syn operator such that

$$B(T_k|d_i) = 0.4 + 0.6 \left\{ \frac{x'_i(T_k)}{x'_i(T_k) + 0.5 + 1.5 l_i/\bar{l}} \times \frac{\log[(N + 0.5)/n'(T_k)]}{\log(N + 1)} \right\}, \tag{2.74}$$

where $x'_i(T_k) = \sum_{j:t_j \in T_k} \mathrm{f}_{ij}$ and $n'(T_k)$ indicates the number of documents including at least one translation in T_k [143, 291, 184, 124]. Pirkola's method has been slightly modified by some studies [74, 311]. For instance, in an experiment [74], translation probabilities are incorporated into the computation of $x'_i(T_k)$ and $n'(T_k)$ such as $x'_i(T_k) = \sum_{j:t_j \in T_k} P(t_j|s_k) \times \mathrm{f}_{ij}$ and $n'(T_k) = \sum_{j:t_j \in T_k} P(t_j|s_k) \times n_j$ in order to increase influence of probable target terms with higher translation probabilities.

2.2.6 Method for multilingual information retrieval

Approach to MLIR

Suppose that there is a multilingual document collection in which two or more languages are mixed (not a parallel corpus), and that a user wishes to search the collection for a query expressed in a single language (see Figure 2.1). This MLIR task is more complicated than simple BLIR. Basically, there are two strategies for MLIR as follows [188]:

- *Distributed architecture* in which the document collection is separated by language, and each part is indexed and retrieved independently.
- *Centralized architecture* in which the document collection in various languages is viewed as a single document collection and is indexed in one huge index file.

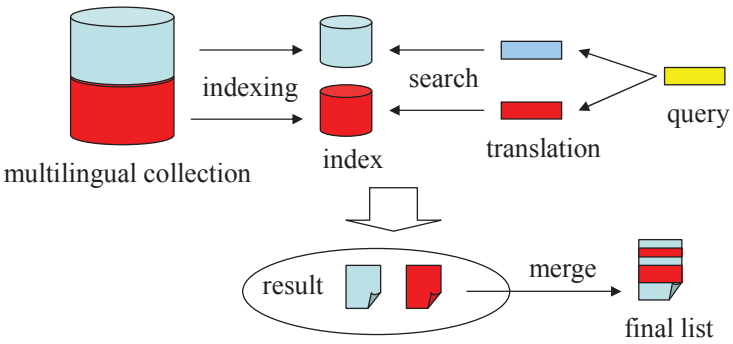


Figure 2.9: Distributed architecture for MLIR
Source: Kishida(2010b)[153]

Merging technique

In the distributed architecture, a standard bilingual search is repeatedly performed for each separate language sub-collection respectively, and several ranked document lists are generated by each run (see Figure 2.9). Then the problem becomes how to merge the results of each run into a single ranked list so that all relevant documents in any language are successfully located on upper ranks. Essentially, the *merging strategy* is a general research topic of IR when searching distributed resources (i.e., distributed IR), in which it is necessary to merge ranked lists obtained from each resource. In CLIR, the following merging strategies have been investigated [39]:

- *Raw score*. Using straightforwardly document scores estimated in each run.

[39] Experimental comparison of the search performance between these methods has been attempted by [276].

- *Round robin*. Interleaving each document list in a round robin fashion by assuming that the distribution of relevant documents is identical among the lists.
- *Normalized score*. Normalizing document scores of each run in order to remove the effects of collection-dependent statistics used for estimating the scores.
- *Rank-based score*. Converting mathematically ranks in each run into scores by assuming a relationship between probabilities of relevance and the ranks.
- *Modified score*. Modifying raw scores in each run so as to reduce the effects of collection-size dependency, translation ambiguity, and so on.

If the retrieval model employed for each run can estimate the relevance probability of each document correctly, then it would be reasonable to re-rank all documents together according to values of the probability (i.e., raw score strategy). For instance, the logistic regression model for IR can be used as a retrieval model for it (see [57] for details).

However, in most cases, it would be too difficult to interpret each document score as a pure probability of relevance even though a probabilistic retrieval model was actually used because the probability is often approximately estimated for convenience of calculation in the model. In such cases, if it can be assumed that relevant documents are distributed in the same way within every separate language sub-collection, then it is possible to employ round robin-based merging in which only the rank of each document is taken into account. Otherwise, an alternative method is to use normalized document scores such that

$$\mathrm{v}_i' = (\mathrm{v}_i - \mathrm{v}_{\min})/(\mathrm{v}_{\max} - \mathrm{v}_{\min}), \tag{2.75}$$

where v_i is the raw score of document d_i, and $\mathrm{v}_{\min}$ and $\mathrm{v}_{\max}$ are the minimum and maximum in each search run respectively [242].

One widely-known method for normalization in distributed IR is CORI (collection retrieval inference) algorithm [48]. In this algorithm, first, the score of a sub-collection for the given query is computed based on Equation (2.73) by considering each sub-collection D_k $(k = 1, ..., K)$ as a single huge document where K is the number of sub-collections included in the multilingual collection (i.e., the number of languages). The score is denoted by $B(q|D_k)$. Second, the score v_i $(= B(q|d_i))$ of a document d_i in a sub-collection D_k (i.e., $d_i \in D_k$) is converted such that

$$\mathrm{v}_i' = \mathrm{v}_i \times \left(1 + K \times \frac{B(q|D_k) - \bar{B}(q)}{\bar{B}(q)}\right), \tag{2.76}$$

where $\bar{B}(q) = K^{-1} \sum_{k=1}^{K} B(q|D_k)$. For more details on applying the CORI algorithm to MLIR, see [274] or [218].

Another possibility is to predict the relevance probability of a document ranked in a position by using training data sets [93, 168, 275]. For instance, it is possible to calculate this probability such that

$$\hat{P}(\mathcal{R}|d_i) = a + b \log(\rho_i), \tag{2.77}$$

where $P(\mathcal{R}|d_i)$ indicates the relevance probability of document d_i, ρ_i denotes its rank in the output list, and a and b are parameters to be estimated from training data (a more complicated regression model was used in [275]). Meanwhile, researchers have explored other techniques of modifying raw scores so as to remove the effect of collection-size dependency (e.g., [129]), or of reducing them based on the degree of translation ambiguity according to the assumption that a good translation may give much more relevant documents (e.g., [188]).

Searching heterogeneous collection

In the centralized architecture, the set of multilingual documents is not divided into sub-collections for each language. For searching such a heterogeneous collection, it is necessary either

1. to translate a given source query into all languages included in the document collection and to merge all translations into a single query, or

2. to translate all documents in a single language used in the query.

In general, the first method has been adopted (e.g., [106, 221]). Since documents in a language having fewer documents may take advantage of weighting by document frequency, it may be necessary in this method to adjust the idf factor [188]. For example, in Equation (2.73), $\{\log[(N+0.5)/n_j]\}/\log(N+1)$ is an idf factor, and its value becomes larger as n_j is smaller.

Meanwhile, in the second method, it is not necessary to add such adjustment of parameters because a single index file registering query language words is created even though the document translation is a very time-consuming task. This method was attempted by [58], and the search performance of various MLIR techniques discussed in this section was experimentally compared.

Chapter 3

Hierarchical and Non-Hierarchical Document Clustering

3.1 Introduction to Document Clustering Technique

In this section, basic elements and procedures of document clustering (DC) are introduced as fundamentals for discussing various DC techniques in the following sections.

3.1.1 Document representation and similarity

Document vector

As similarly to information retrieval (IR), in DC or text categorization, a document is usually represented by a vector whose elements are term weights such as $\mathbf{d}_i = [w_{i1}, w_{i2}, ..., w_{iM}]^T$ where w_{ij} indicates a weight of term t_j in d_i ($i = 1, \ldots, N$; $j = 1, \ldots, M$) and M is the total number of distinct terms in D as before. Note that 'term' means an index term extracted by an indexing process from the original text of each document (see Section 2.1.6). The simplest weighting scheme is that $w_{ij} = 1$ if t_j appears in d_i, and if not, $w_{ij} = 0$, by which a *binary vector* is constructed for each document. If term frequencies are straightforwardly used, then $w_{ij} = \mathrm{f}_{ij}$ where f_{ij} is the times that t_j appears in d_i. Otherwise, various methods of tf-idf weighting can be applied for defining w_{ij} (see Section 2.1.3).

Suppose that there is a sample document database shown in Table 3.1 (for convenience, Table 3.1 is called 'sample DB' in this thesis) where n_j indicates *document frequency* of term t_j (i.e., the number of documents including term t_j), and l_i means *document length* of document d_i (i.e., $l_i = \sum_{j=1}^{M} \mathrm{f}_{ij}$), as before. A document vector of d_1 in the sample DB can be computed as follows:

- Binary vector. $\mathbf{d}_1 = [1, 1, 0, 0, 0, 0]^T$.
- Term frequency (tf) vector. $\mathbf{d}_1 = [4, 1, 0, 0, 0, 0]^T$.

Table 3.1: Sample document database ('sample DB')

		term frequency f_{ij}									
t_j	n_j	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
t_1	3	4	1	1	0	0	0	0	0	0	0
t_2	6	1	3	6	0	2	1	1	0	0	0
t_3	4	0	0	0	4	2	1	5	0	0	0
t_4	4	0	0	0	3	0	0	0	2	1	3
t_5	6	0	0	0	0	1	5	1	3	5	1
t_6	2	0	0	0	0	0	0	1	0	0	4
l_i	-	5	4	7	7	5	7	8	5	6	8

Table 3.2: Values of idf: $\log N/n_j$

n_j/N	0.01	0.02	0.05	0.10	0.20	0.50	0.80	1.00
$\log_e N/n_j$	4.61	3.91	3.00	2.30	1.61	0.69	0.22	0.00
$\log_2 N/n_j$	6.64	5.64	4.32	3.32	2.32	1.00	0.32	0.00
% for $\log(0.01)^{-1}$	100%	85%	65%	50%	35%	15%	5%	0%

Also, tf vectors are often normalized explicitly such that $w_{ij} = \mathrm{f}_{ij}/l_i$, namely,

- Normalized tf vector: $\mathbf{d}_1 = [4/5, 1/5, 0, 0, 0, 0]^T$,

because $l_1 = 4 + 1 = 5$.

Furthermore, an example of vector based on tf-idf weighting is

- Vector by tf-idf weighting, $\mathrm{f}_{ij} \times \log(N/n_j)$: $\mathbf{d}_1 = [4.8, 0.5, 0.0, 0.0, 0.0, 0.0]^T$.

Table 3.2 may help us understand influence of the idf factor on term weighting. In the case of a simple formula that $\mathrm{f}_{ij} \times \log(N/n_j)$, each counted tf is changed by multiplying an idf value like those in the table. As a result, weights of terms appearing in a smaller part of the entire set increase in comparison with those included in many more documents (note that n_j/N indicates a portion of documents including term t_j in the set). Thus by incorporating the idf factor, importance of specific terms intensively discussed in a particular set of documents (e.g., "clustering") is expected to augment more than that of general terms widely used in various documents (e.g., "research"), and actually, the idf factor is indispensable for document ranking in IR. For computing document similarities in DC, the tf-idf weighting would be effective in some situations.

Generally in DC, the number of terms in a document collection, M, is so large and the vectors have high-dimensionality [1]. Also, many zero elements appear in the vectors because only a limited part of M terms is actually included in each document. Consequently, the term-by-document matrix such as Table 3.1 becomes very *sparse*, which often prevents standard classification techniques from being simply applied to DC or text categorization problems.

Similarity measure

In DC, it is usually necessary to compute the degree of a topical *similarity* between two document vectors or between a document vector and a cluster vector that is a subject representation of each cluster. The similarity between $\mathbf{d}_i$ and $\mathbf{d}_h$ can be measured as

- Cosine coefficient: $\sum_{j=1}^{M} w_{ij} w_{hj} \Big/ \sqrt{\sum_{j=1}^{M} w_{ij}^2 \sum_{j=1}^{M} w_{hj}^2}$, or
- Euclidean distance: $\sqrt{\sum_{j=1}^{M} (w_{ij} - w_{hj})^2}$.

For instance, if tf vectors are adopted, then the cosine coefficient between $\mathbf{d}_1$ and $\mathbf{d}_2$ in Table 3.1 is computed as

$$\frac{4 \times 1 + 1 \times 3}{\sqrt{4^2 + 1^2} \times \sqrt{1^2 + 3^2}} = \frac{7}{4.123 \times 3.162} = 0.537.$$

Also its Euclidean distance becomes $\sqrt{(4-1)^2 + (1-3)^2} = \sqrt{9+4} = 3.606$. Note that a larger cosine value means higher similarity between two documents whereas a larger distance indicates lower similarity between them, inversely.

According to the vector space model (Section 2.1.3), DC techniques are usually based on the cosine coefficient, rather than the Euclidean distance that is widely used in other fields. Like Equation (2.12), the cosine coefficient between two document vectors is geometrically written as

$$\cos\theta = \frac{\mathbf{d}_i^T \mathbf{d}_h}{\|\mathbf{d}_i\| \, \|\mathbf{d}_h\|} \tag{3.1}$$

[1] Actually, M can be reduced by selecting a subset from all terms appearing in the set D as features for classification, as discussed later. The *feature* means generally an attribute or a characteristic of target objects to be classified.

where θ is an angle between two vectors and $\|\cdot\|$ represents a vector *norm* (e.g., $\|\mathbf{d}_i\| = \sqrt{\mathbf{d}_i^T \mathbf{d}_i}$). When all weights in document vectors are non-negative, $0 \leq \cos\theta \leq 1$ (if $\theta = 0°$ then $\cos\theta = 1$ whereas if $\theta = 90°$ then $\cos\theta = 0$). By using normalized vectors whose is 1 (it is generally called *unit vector*) such as

$$\tilde{\mathbf{d}}_i = \mathbf{d}_i / \|\mathbf{d}_i\|, \tag{3.2}$$

the cosine coefficient in Equation (3.1) can be written simply as $\tilde{\mathbf{d}}_i^T \tilde{\mathbf{d}}_h$ where $\|\tilde{\mathbf{d}}_i\| = \|\tilde{\mathbf{d}}_h\| = 1$. This means that document length has no effect on the degree of cosine measure unlike the Euclidean distance.

For example, in Figure 3.1, three tf vectors of documents d_A, d_B and d_C are plotted in a space with two dimensions corresponding to terms t_1 and t_2. Since tf vector of d_A is longer, the Euclidean distance between d_A and d_B is larger than that between d_B and d_C, which means lower similarity between d_A and d_B. In contrast, values of the cosine coefficient lead to a reverse conclusion since $\cos\theta > \cos\phi$.

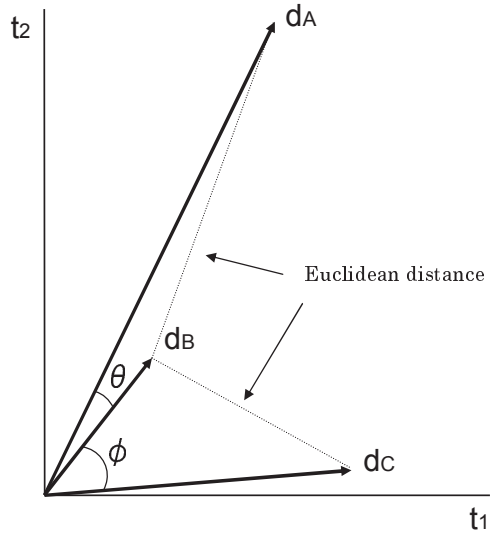


Figure 3.1: Document vectors in two-dimensional space

Modification of term weights by the idf factor is straightforwardly reflected on values of the cosine coefficient. For example, its value between $\mathbf{d}_2$ and $\mathbf{d}_3$ in the sample DB (Table 3.1) reduces slightly to 0.96 from 0.99 by changing term weights from simple tf to $w_{ij} \times \log N/n_j$ partly because t_2 appearing frequently in both the documents is also included in many documents (i.e., $n_2 = 6$), and its weight becomes relatively lower [2].

Use of unit vector

A unit vector other than Equation (3.2) can be obtained as

$$\tilde{\mathbf{d}}_i = [\sqrt{\mathrm{f}_{i1}/l_i}, \ldots, \sqrt{\mathrm{f}_{iM}/l_i}]^T, \tag{3.3}$$

based on the normalized tf vector. For example, vectors of document d_1 in Table 3.1 becomes as follows:

- Equation (3.2). $\mathbf{d}_1 = [0.970, \ldots, 0.242, \ldots, 0.0, 0.0, 0.0, 0.0]^T$.
- Equation (3.3). $\mathbf{d}_1 = [0.941, \ldots, 0.058, \ldots, 0.0, 0.0, 0.0, 0.0]^T$.

[2] For the cosine measure, $\log_e$ and $\log_2$ in the idf factor bring the same value because $\log_e x = \log_2 x / \log_2 e$ where $\log_2 e$ is common among all terms and canceled finally.

In the case of using these unit vectors, since

$$\begin{aligned} \|\tilde{\mathbf{d}}_i - \tilde{\mathbf{d}}_h\|^2 &= \sum_j (w_{ij} - w_{hj})^2 = \sum_j w_{ij}^2 - 2\sum_j w_{ij}w_{hj} + \sum_j w_{hj}^2 \\ &= 1 - 2\sum_j w_{ij}w_{hj} + 1 = 2 - 2\tilde{\mathbf{d}}_i^T \tilde{\mathbf{d}}_h, \end{aligned} \tag{3.4}$$

the square of the Euclidean distance brings the same clustering result with that by the cosine coefficient [278]. Unit vectors of d_1, d_2 and d_3 in Table 3.1 can be plotted on a unit circle shown as Figure 3.2. Such kind of observations on a circle or a sphere can be treated by the theory of directional statistics [203], which is discussed in Section 4.1.1.

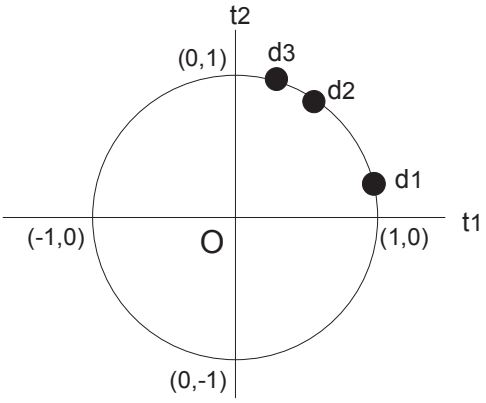
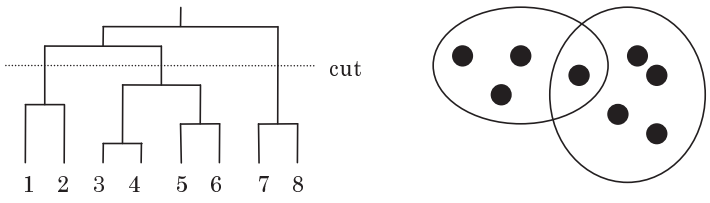


Figure 3.2: Unit vectors on circle

3.1.2 Types of document clustering

Hierarchical and non-hierarchical clustering

Clustering can be generally grouped into two types based on their forms of clustering results, namely, hierarchical and non-hierarchical clustering (see Figure 3.3). In *hierarchical clustering*, a set of nested clusters is generated. For example, cluster $\{3, 4\}$ and cluster $\{5, 6\}$ are merged into a nested cluster $\{\{3, 4\}, \{5, 6\}\}$ in Figure 3.3. Consequently, it is feasible to obtain a tree structure of nested clusters, which is usually called *dendrogram*. Therefore, for producing a set of clusters defined in Equation (1.1), the dendrogram has to be cut at a level. In Figure 3.3, three clusters, $\{1, 2\}$, $\{3, 4, 5, 6\}$ and $\{7, 8\}$, are created by a cut of dendrogram.



(a) Hierarchical

(b) Non-hierarchical (nonexclusive)

Figure 3.3: Document vectors in two-dimensional space

By *non-hierarchical clustering* which is often called *flat partitioning*, a non-nested structure as shown in Figure 3.3 is obtained, which leads to directly a set of clusters in Equation (1.1). Because

Table 3.3: Similarity matrix of sample DB

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
d_2	.54								
d_3	.40	.99							
d_4	.00	.00	.00						
d_5	.16	.63	.66	.53					
d_6	.05	.18	.19	.15	.58				
d_7	.05	.18	.19	.76	.82	.40			
d_8	.00	.00	.00	.33	.28	.80	.16		
d_9	.00	.00	.00	.12	.33	.94	.19	.92	
d_{10}	.00	.00	.00	.35	.07	.19	.19	.49	.31

computational complexity of hierarchical clustering is fairly high as discussed later, it is hard to apply the algorithm to a large-scale document set. In the IR field, for solving the problem, a *single-pass algorithm* creating a flat partition has been often employed for DC where 'single-pass' means that clusters can be generated by only a single scan of the document file, and as a result, computational complexity decreases largely.

Outline of hierarchical clustering

In the case of *agglomerative* hierarchical clustering, it is necessary to compute a *similarity matrix* $\mathbf{S} = [s_{ih}]$ $(i, h = 1, \ldots, N)$ where s_{ih} denotes the degree of similarity between d_i and d_h. Table 3.3 is an example of the similarity matrix in which each element is a value of the cosine coefficient calculated from tf vectors in the sample DB. Because $\mathbf{S}$ is symmetric (i.e., $s_{ih} = s_{hi}$) and always $s_{ii} = 1.0$ in the case of using the cosine coefficient, Table 3.3 displays $\mathbf{S}$ as a lower triangle matrix removing diagonal elements.

Based on the set of s_{ih}, it is possible to define the degree of similarity between clusters $\tilde{s}_{km}$ $(k, m = 1, \ldots, L; k \neq m)$ such that

- *single linkage*: $\tilde{s}_{km} = \max\{s_{ih} | d_i \in C_k, d_h \in C_m\}$,
- *complete linkage*: $\tilde{s}_{km} = \min\{s_{ih} | d_i \in C_k, d_h \in C_m\}$,
- *group average linkage (UPGMA)*: $\tilde{s}_{km} = (\tilde{n}_k \times \tilde{n}_m)^{-1} \sum_{i:d_i \in C_k} \sum_{h:d_h \in C_m} s_{ih}$,

where $\tilde{n}_k$ and $\tilde{n}_m$ are the number of documents belonging to C_k and C_m, respectively (e.g., $\tilde{n}_k = |C_k|$). Also, $\min\{\cdot\}$ and $\max\{\cdot\}$ represent minimum and maximum of elements in the set $\{\cdot\}$, respectively. While the three definitions have been often used for DC, there are other types of similarity (distance) between clusters generally [3].

Figure 3.4 shows an image of single linkage, complete linkage and group average linkage for defining similarities between clusters (exactly, representation of the figure is based on distance, not similarity). Whereas the similarity between the nearest pair is used as cluster similarity in the single linkage method, the complete linkage method adopts the similarity between the farthest pair. In group average linkage, an average of similarities between all pairs is employed for avoiding a singular effect from a particular nearest or farthest pair.

A dendrogram can be constructed by merging repeatedly two clusters whose similarity is the highest in the current stage. In the process, a single document has to be considered as a pseudo cluster, which is usually called *singleton*, unless the document enters into any cluster. For example, when the single linkage method is applied to data of Table 3.3, document d_6 is combined with d_9 into a cluster (its similarity is 0.94), and next, singleton d_8 is merged with cluster $\{d_6, d_9\}$ (its similarity is 0.92 because 0.92 between d_8 and d_9 is larger than 0.80 between d_6 and d_8). Furthermore, the cluster $\{d_6, d_8, d_9\}$ is merged with $\{d_2, d_3, d_4, d_5, d_7\}$ (its similarity is 0.40 between d_6 and d_7). Note that the similarity of d_8 and $\{d_6, d_9\}$ is 0.80, not 0.92, in the case of complete linkage.

[3] For instance, a textbook on clustering [322] introduces also 'weighted average linkage (WPGMA)', 'median linkage (WPGMC)', 'centroid linkage (UPGMC)' and Ward's method.

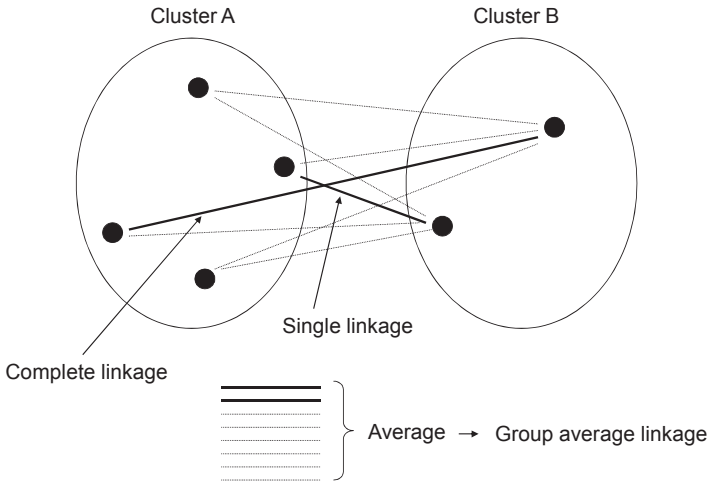
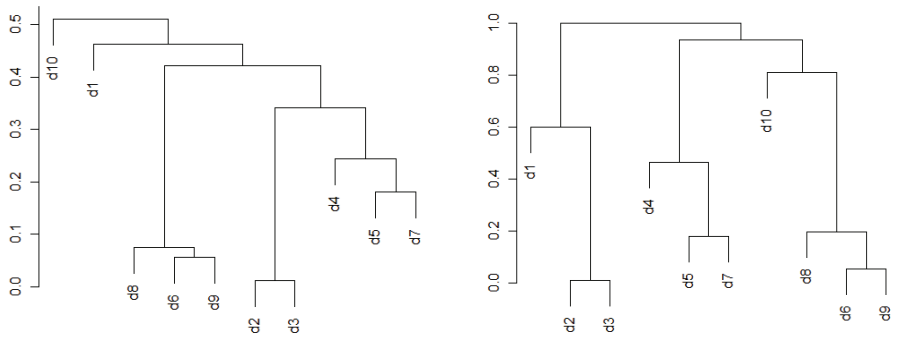


Figure 3.4: Distance between clusters

Actually, Figure 3.5 shows dendrograms created by the single linkage and complete linkage methods from data in Table 3.3. The dendrograms were obtained by hclust() function of a statistical language R [246] and the degrees of similarity was converted into distance measures as $1-s_{ih}$ before input of data to R (see scale indicators of the left-hand side in each dendrogram). Interestingly, it seems that dendrogram by the complete linkage is more valid than that by the single linkage. For example, even though d_1 shares terms with only d_2 and d_3, d_1 is located far from the cluster including d_2 and d_3 in the dendrogram by the single linkage. Such kind of unsuccessful result by the single linkage is often pointed out in literature, and in particular, it is well-known that the single linkage method may generate an unbalanced tree such that in Figure 3.6 by so-called 'chain-effect'.

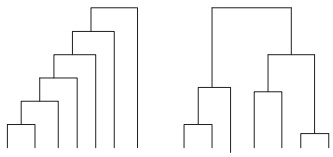


(a) single linkage (b) complete linkage

Figure 3.5: Clustering results for sample DB by R-2.12.0

When the tree by the complete linkage is cut at the level of 0.7 (i.e., 0.3 of the cosine coefficient), four clusters can be obtained (i.e., $C_1 = \{d_1, d_2, d_3\}$, $C_2 = \{d_4, d_5, d_7\}$, $C_3 = \{d_6, d_8, d_9\}$ and $C_4 = \{d_{10}\}$). If the level is changed to 0.85 (i.e., 0.15 of the cosine coefficient), then d_{10} is merged into cluster C_3.

By adopting unit vectors, other clustering methods specialized to the Euclidean distance can be also used for DC. For instance, a result from applying the Ward's method to the sample DB based on the Euclidean distance between unit vectors computed from tf vectors by Equation (3.2) is shown as Figure 3.7, which also contains a dendrogram obtained by using original tf vectors. It seems that the dendrogram by the Ward's method based on unit vectors is very similar with



(a) Unbalanced tree (b) Balanced tree

Figure 3.6: Unbalanced and balanced trees

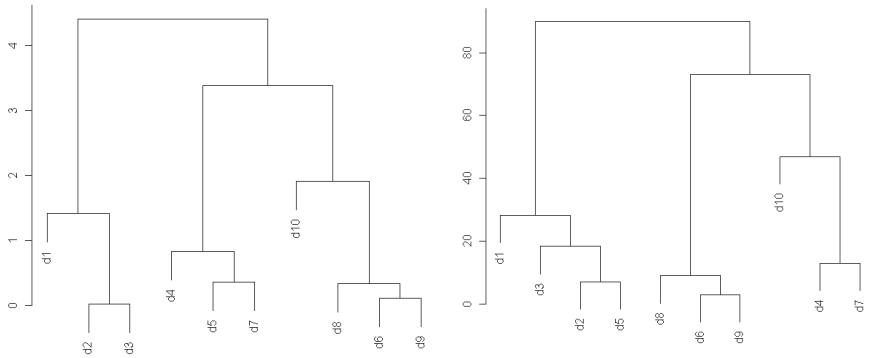
that by the complete linkage, whereas the original tf vectors would create an inappropriate tree in which d_5 is combined with d_2 in the earlier stage. The Ward's method tries to find two clusters to be merged in each stage based on *sum of squared error* or *residual sum of squares* (RSS) of a cluster,

$$J_k = \sum_{i:d_i \in C_k} \|\mathbf{d}_i - \mathbf{m}_k\|^2, \tag{3.5}$$

where

$$\mathbf{m}_k = \frac{1}{\tilde{n}_k} \sum_{i:d_i \in C_k} \mathbf{d}_i, \tag{3.6}$$

which is called *centroid vector* in the cluster [9]. If a new cluster C_p is created by merging C_k with C_m, then increment of the sum of squared error caused by creating C_p can be computed as $\triangle J_{km} = J_p - J_k - J_m$. In the Ward's method, $\triangle J_{km}$ is used as distance measure between C_k and C_m, and the nearest two clusters are merged in each stage [4].



(a) Unit vectors (b) Original tf vectors

Figure 3.7: Clustering results by Ward's method using R-2.12.0 (left:$\mathbf{d}_i/\|\mathbf{d}_i\|$, right: $\mathbf{d}_i$)

Tree structure obtained by hierarchical clustering such as shown in Figures 3.5 and 3.7 is often useful for IR applications. For instance, when a search engine tries to divide a result for a given query into topically homogeneous subsets, it would be convenient for the user to adjust the number of documents by moving up and down in the tree. Unfortunately, in order to execute agglomerative hierarchical clustering, similarity scores between $O(N^2)$ pairs have to be computed, which often prevents us from applying hierarchical methods to a large set of documents. For example, whereas Table 3.3 includes only 45 values of the cosine coefficient, when $N = 10000$, the number of pairs amounts to 49,995,000 [5]. In addition, a pair of clusters to be merged in each stage has to be found, in which numerous checks of values in the matrix are needed (see Section 3.2). Thus flat partitioning techniques in which computation of the similarity matrix is not required have been often used in the IR field as mentioned above.

[4]The distance is computed such that $\triangle J_{km} = \tilde{n}_k \tilde{n}_m / (\tilde{n}_k + \tilde{n}_m) \times \|\mathbf{m}_k - \mathbf{m}_m\|^2$ [9].
[5]The number of pairs can be calculated as $(N \times (N-1))/2$.

Outline of non-hierarchical clustering

The most typical non-hierarchical clustering method is possibly *k-means* algorithm where 'k' indicates the number of clusters, which has to be determined a priori before executing the algorithm. In this thesis, the number of clusters 'k' is denoted by L for keeping consistency of notations. A basic procedure of k-means algorithm is shown in Figure 3.8.

‡ Basic k-means algorithm (batch mode)

Set: The number of clusters L, and initial cluster vectors, $\mathbf{c}_1, ..., \mathbf{c}_L$.

1) Allocate document d_i to the nearest cluster ($i = 1, \ldots, N$).

2) Update cluster vectors $\mathbf{c}_k$ based on the allocation in step 1) ($k = 1, \ldots, L$).

3) Terminate the algorithm if each $\mathbf{c}_k$ becomes stable. Otherwise, return to step 1).

Out: Clusters $C_1, \ldots, C_L$ and final vectors $\mathbf{c}_1, ..., \mathbf{c}_L$.

Figure 3.8: Basic k-means algorithm

Cluster vectors are defined as

$$\mathbf{c}_k = [\tilde{w}_{k1}, \tilde{w}_{k2}, \ldots, \tilde{w}_{kM}]^T, \ k = 1, \ldots, L, \tag{3.7}$$

where $\tilde{w}_{kj}$ means a weight of term t_j in C_k. Typically, the centroid vector in Equation (3.6) is adopted as the cluster vector (i.e., $\mathbf{c}_k = \mathbf{m}_k$) in k-means algorithms. One of the methods for setting the initial cluster vector is to select randomly a document, which is generally called *seed*, for C_k ($k = 1, \ldots, L$). When the random selection is applied, it is enough to specify only the number of clusters L before executing the algorithm.

After the seeds are determined, each document vector is sequentially checked and allocated to the nearest cluster, whose distance to the document is the shortest. Next, each cluster vector is updated based on the result of allocations. In the case that $\mathbf{c}_k = \mathbf{m}_k$, each centroid vector is simply recomputed for documents allocated to each cluster at the stage. When all cluster vectors are updated together after allocations of all documents as in Figure 3.8, it is usually called *batch mode* k-means clustering. Meanwhile, in the case of 'incremental' or 'online mode' k-means clustering, after a single document is allocated to a cluster, the related cluster vectors (i.e., those of the old and new clusters for the document) are immediately updated.

The allocating and updating processes are usually repeated until cluster vectors are not changed (see Figure 3.9). Otherwise, the algorithm can be stopped by checking RSS for the entire set of clusters, which is defined as

$$J = \sum_{k=1}^{L} J_k = \sum_{k=1}^{L} \sum_{i:d_i \in C_k} \left\| \mathbf{d}_i - \frac{1}{\tilde{n}_k} \sum_{i:d_i \in C_k} \mathbf{d}_i \right\|^2 \tag{3.8}$$

(see Equations (3.5) and (3.6)). Namely, when the RSS becomes lower than a threshold or the amount of decrease in the RSS falls under a threshold, the algorithm may be terminated [202].

Table 3.4 shows a result of the basic k-means clustering for the sample DB when the number of clusters are given as $L = 2, 3, 4$, respectively. Note that unit vector $\mathbf{d}_i / \|\mathbf{d}_i\|$ computed from tf vector was used in the input data and that sets of documents, $\{d_5, d_6\}$, $\{d_5, d_6, d_7\}$ and $\{d_5, d_6, d_7, d_8\}$, were particularly selected as seeds for each execution [6]. In the case that $L = 3$, this algorithm

[6] From theoretical point of view, it may be better to normalize centroid vectors such that $\tilde{\mathbf{m}}_k = \mathbf{m}_k / \|\mathbf{m}_k\|$ when measuring distance between document and cluster representations in DC. Another strategy is to consider the unit vectors of documents as representations from which influence of document length is removed, and to treat them simply as data entering into a clustering algorithm without any consideration of normalizing centroid vectors. This chapter adopts basically the latter strategy, which enables to straightforwardly employ standard techniques or software packages for k-means clustering, whereas in Section 4.1.1, an algorithm based on normalized cluster vectors will be also discussed.

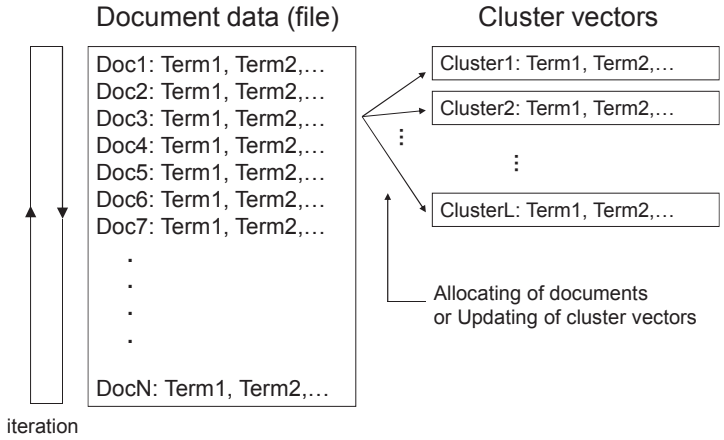


Figure 3.9: Outline of k-means clustering algorithm

Table 3.4: Clusters and their vectors generated by basic k-means algorithm

		Cluster vectors					
Clusters	Documents	t_1	t_2	t_3	t_4	t_5	t_6
$L = 2$ (seeds: d_5, d_6)							
C_1	$d_1, d_2, d_3, d_4, d_5, d_7$	.242	.506	.402	.100	.087	.031
C_2	d_6, d_8, d_9, d_{10}	.000	.048	.048	.335	.743	.196
$L = 3$ (seeds: d_5, d_6, d_7)							
C_1	d_1, d_2, d_3	.484	.726	.000	.000	.000	.000
C_2	d_4, d_5, d_7	.000	.285	.804	.200	.174	.063
C_3	d_6, d_8, d_9, d_{10}	.000	.048	.048	.335	.743	.196
$L = 4$ (seeds: d_5, d_6, d_7, d_8)							
C_1	d_1, d_2, d_3	.484	.726	.000	.000	.000	.000
C_2	d_4, d_5, d_7	.000	.285	.804	.200	.174	.063
C_3	d_6, d_8, d_9	.000	.064	.064	.250	.925	.000
C_4	d_{10}	.000	.000	.000	.588	.196	.784

generated $C_1 = \{d_1, d_2, d_3\}$, $C_2 = \{d_4, d_5, d_7\}$ and $C_3 = \{d_6, d_8, d_9, d_{10}\}$, which are explicitly consistent with results by the complete linkage and the Ward's methods (see Figures 3.5 and 3.7). Similarly, it is easy to grasp that the sets of clusters obtained for $L = 2$ and $L = 4$ in Table 3.4 can be correctly extracted from dendrograms of Figures 3.5 and 3.7 by setting appropriate levels of tree cut.

Unfortunately, inappropriate clusters may be generated by other seeds. For example, the basic k-means algorithm with the seeds $\{d_1, d_2, d_5\}$ provides three clusters

$$C_1 = \{d_1\}, C_2 = \{d_2, d_3\}, C_3 = \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}.$$

Thus when seeds are randomly selected, it is possible that invalid clustering results are caused by the selected seeds [7]. More sophisticated k-means algorithms reducing possibility that inappropriate clusters are created will be explained in Section 3.3.1.

If the total number of iterations is denoted by r, then computational complexity of the k-means clustering can be represented by $O(NLMr)$ because $N \times L$ matching operations between M-dimensional vectors are required in each iteration. Thus the k-means algorithm is expected to be sufficiently faster than agglomerative hierarchical methods if L and r are enough small. In

[7] It should be noted that clustering result of the basic k-means algorithm is affected by the order of documents to be processed as well as the selected seeds. The sets of clusters shown in Table 3.4 were obtained by processing documents from d_1 to d_{10} in turn. If the other order is used, then different clusters from those in Table 3.4 may be generated.

an extreme case where $r = 1$, a clustering result can be obtained by only a single scan of the target document file (i.e., single-pass clustering) although its quality may not be high. So-called 'leader-follower clustering algorithm' is another method often used for single-pass clustering of large document sets (see Section 3.3.2). Unfortunately, when $L \times r \simeq N$, it becomes that $O(NLr) \simeq O(N^2)$, which means that the computational complexity of k-means clustering approaches to that of agglomerative hierarchical methods even though k-means clustering does not require calculation of a similarity matrix.

Other than the k-means and leader-follower algorithms, probabilistic clustering, matrix-based clustering, graph-based clustering (see Chapter 4) and so on can be categorized into the type of non-hierarchical clustering (i.e., flat partitioning).

Remarks on link-based clustering

Similarity between two documents can be also measured based on citation links or web links. For instance, bibliometric studies often use *co-citation* measure for visualizing relationships between scientific papers in a research field. Suppose that there are 10 papers referring concurrently to both d_1 and d_2. In this case, the co-citation counts of d_1 and d_2 amount to 10, which is denoted by $\mathrm{c}_{12} = 10$ here (i.e., these 10 papers are *citing* documents, and d_1 and d_2 are *cited* documents). If the total numbers of documents citing d_1 and d_2 are written as c_1 and c_2, respectively, then the cosine coefficient is computed by $\mathrm{c}_{12}/\sqrt{\mathrm{c}_1\mathrm{c}_2}$, from which a similarity matrix like Table 3.3 can be constructed. Also, other coefficients such as the Dice coefficient $2\mathrm{c}_{12}/(\mathrm{c}_1 + \mathrm{c}_2)$, the Jaccard coefficient $\mathrm{c}_{12}/(\mathrm{c}_1 + \mathrm{c}_2 - \mathrm{c}_{12})$, the overlap coefficient $\mathrm{c}_{12}/\min(\mathrm{c}_1, \mathrm{c}_2)$ and so on, can be employed.

Recently, *link-based* clustering [214] becomes more important after the web is prevalent because hyperlinks embedded into the web pages can be used like citation links. The hyperlink information allows for grouping web sites into clusters within which a close relationship is maintained. A specific network structure existed in a set of web sites may be uncovered by the clustering (e.g., see [303]). Such clustering methods using link relationship between documents should be technically discerned from *text-based* clustering [8] which is executed by using text information included in documents.

Another technique of link-based document clustering would be to improve results of text-based clustering by using information contained in 'neighbors' linked with each document. For instance, the cluster to which the target document was assigned by a unsupervised manner can be changed into another cluster based on assignments of its neighbors [9]. However, this thesis focuses on only text-based clustering that utilizes term information contained in documents, and the link-base clustering by bibliographic links or hyperlinks on the web is out of the scope.

3.1.3 Feature selection

Heaps' law and Zipf's law

As mentioned repeatedly, document sets are often very large, which may prevent us from executing DC in real situations. The difficulty of DC is accelerated by another fact that dimension of cluster vectors becomes higher as the document set is larger (i.e., the value of M is often so big). Also, a set of terms appearing in a document is usually so different from that of another one [10], which leads to a very sparse term-by-document matrix as pointed out in Chapter 1.

A well-known *Heaps' law* [122, 202] allows to predict the vocabulary size of a document set such as

$$M = aC^b \tag{3.9}$$

where C indicates 'collection size' (or *collection length*), which is usually measured by the total number of tokens in the document set (i.e., $C = \sum_{i=1}^{N} \sum_{j=1}^{M} \mathrm{f}_{ij}$), and a and b are constants here

[8] Also, it is sometimes called 'content-based' clustering.

[9] A method based on Markov random field (MRF) for labeling clusters has been used to change the clusters in some studies [13, 335].

[10] Note that this makes it difficult to apply *k-medoids* algorithm to DC problems. In the case of the k-medoids algorithm, a single data point (i.e., a document) is selected as a cluster representation, which would not work well if intersections of term sets between a medoid and other documents are small.

$(0 < b < 1)$. According to an estimation by Manning et al.(2008) [202] using RCV1 (Reuters Corpus Volume 1) data [187], it amounts that $\hat{a} = 44$ and $\hat{b} = 0.49$ (see Figure 3.10).

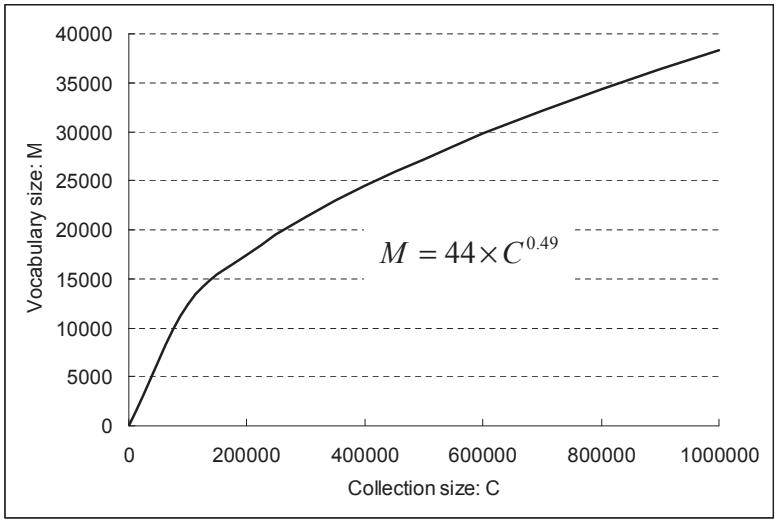


Figure 3.10: Heaps' law

As Figure 3.10 shows, when the document set becomes larger, increasing rate that a new term emerges is slightly reducing. However, the size of vocabulary does not converge to a constant, which means that novel features (i.e., terms) appear usually when a new document is added. Namely, M would not be constant in many cases of DC.

Of course, it may not be necessary to use all terms included in the document set D as features for DC. For instance, terms appearing in only a single document (i.e., $n_j = 1$) have almost no contribution to measurement of similarities between documents (such terms have effect only on calculation of document norm, not on inner product, in the cosine coefficient). Usually, in a document set (or database), there are many terms whose document frequency is only one or a few.

For instance, Figure 3.11 shows an actual distribution of index terms by document frequency, which was compiled from a subset of RCV1 [187] (only about 23,000 news articles published in August 1996). In the distribution, about 43% of all index terms obtained by removing stopwords and applying the Porter's stemming algorithm (Section 2.1.6) is appearing in just one document (where M is about 74,000). These terms may be deleted in the process of computing the similarity degree between document pairs.

The skew distribution in Figure 3.11 has a close relationship with well-known *Zipf's law* [343],

$$x(r) = \frac{A}{r^{\alpha}}, \quad r = 1, 2, \ldots, M, \tag{3.10}$$

where r indicates rank of a term in descending order of occurrence frequency within a document (note that r does not indicate the total number of iterations here) and $x(r)$ denotes the occurrence frequency of r-th term (A and α are constants). By transforming *rank distribution* in Equation (3.10) into a standard frequency distribution $f(x)$ of terms by the number of occurrences x like that in Figure 3.11, it follows that

$$f(x) = \frac{B}{x^{\beta+1}}, \quad x = 1, 2, \ldots \tag{3.11}$$

where $\beta = 1/\alpha$ and $B = \beta A^{\beta}$. Equation (3.11) also mathematically represents the Zipf's law [11].

[11] Rank of a term with x document frequencies can be easily computed from the frequency distribution $f(x)$ such that $r(x) = M + 1 - \sum_{x'=1}^{x} f(x')$. Thus if the summation is approximated by an integral, then

$$r(x) \simeq 1 + M - \int_1^x f(x')dx'.$$

By differentiating both the sides of this equation, it becomes that $f(x) \simeq -dr(x)/dx$. Because $r = (A/x)^{1/\alpha}$ from Equation (3.10), $dr(x)/dx = -\beta A^{\beta} x^{-\beta-1}$ where $\beta = 1/\alpha$, from which Equation (3.11) is derived [145]. Note that when $f(x) = x^c$, $df(x)/dx = cx^{c-1}$ and that $df(kx)/dx = k \times df(x)/dx$ where c and k are constant.

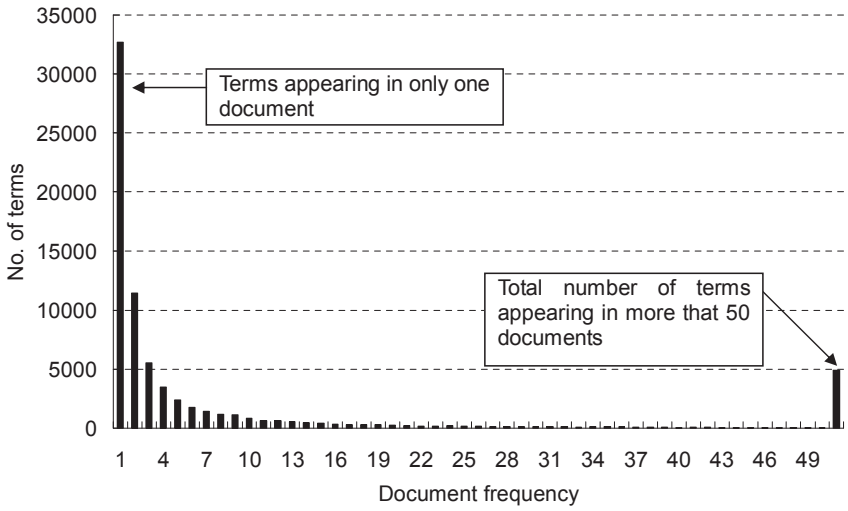


Figure 3.11: Distribution of index terms by document frequency: RCV1 data (August in 1996)

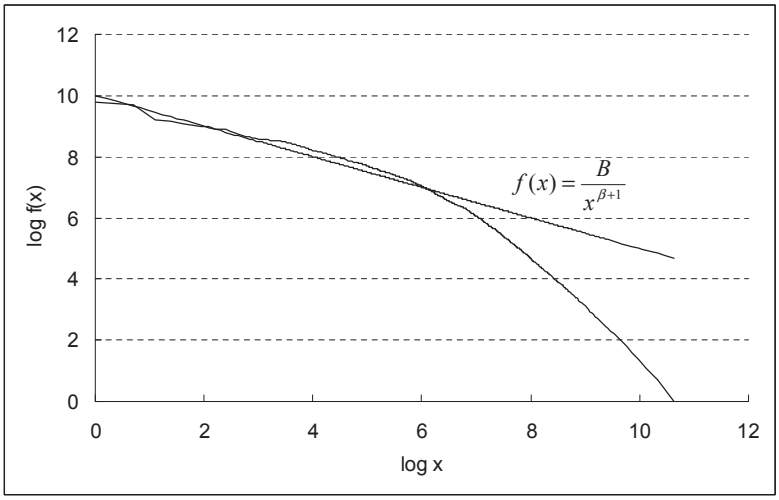


Figure 3.12: Log-log plot of term distribution in Figure 3.11

Figure 3.12 shows a log-log plot of distribution in Figure 3.11, namely, $\log f(x) = \log B - (\beta + 1) \log x$. The straight line in the figure is Equation (3.11) with $\beta = -0.5$ and $B = e^{10} = 22026.4$. Note that x-axis of the distribution in Figure 3.11 is measured by document frequency, not term occurrence frequency in a document which is the original target of Zipf's law, and that the data are not of natural English words but index terms obtained from an indexing process.

Even if terms appearing in only a single document are discarded from a set of features for DC, many terms would still remain, which leads to high-dimensional document vectors again. For example, in the case of Figure 3.11, against about 23,000 document, there are over 40,000 terms whose document frequency is more then one (i.e., $M > N$). For alleviating this problem, it may be better to select a smaller subset of terms as features for classification by identifying usefulness terms for DC.

Feature selection

In text categorization, which is automatic classification of documents in supervised manner, a subset of terms used for the classification can be selected by using the training set, which is called *feature selection*. When documents are classified into two classes (e.g., Classes 1 and 2), the feature selection is not difficult by compiling a contingency table like Figure 2.1 (i.e., 'relevant' and 'irrelevant' are replaced by 'Class 1' and 'Class 2', respectively) from the training dataset. Typically, mutual information or χ^2 statistics have been employed for measuring the usefulness of each term in text categorization (e.g., see [202]).

Meanwhile, for DC in unsupervised manner, such information on the usefulness of terms is not available, and therefore, it would be not easy to apply feature selection to DC problems. One solution may be to rank terms in a document according to their tf-idf weights (Section 2.1.3), and to select some top-ranked terms as features. Of course, the total number of distinct terms M does not always decrease drastically by this method because it is executed for respective document independently (i.e., M can not be controlled directly).

In order to control size of M, it is necessary to find out $M' \times M$ matrix $\mathbf{B}$ such that

$$\bar{\mathbf{d}}_i = \mathbf{B}\mathbf{d}_i, \quad i = 1, \ldots, N, \tag{3.12}$$

where $\mathbf{d}_i$ is an original M-dimensional vector and $M' < M$. As a result, the original document vector can be converted into $\bar{\mathbf{d}}_i$ whose dimension is lower. For computing matrix $\mathbf{B}$, it may be possible to apply LSI-based technique (see Section 2.2.2). Otherwise, as a more efficient method for large-scale document set, matrix $\mathbf{B}$ was randomly generated in experiments of WEBSOM (Section 4.2.1) [141, 166].

3.1.4 Evaluation of clustering results

Types of evaluation

Methods for evaluating clustering results (in other words, for checking 'cluster validity') can be categorized as follows:

1. Direct evaluation
   (a) External evaluation using an 'answer'
   (b) Internal evaluation
2. Indirect evaluation

In the case of indirect evaluation, performance of the application system in which DC is included as a component is assessed. For instance, if a distributed IR system uses a DC module for dividing the target document set into topically homogeneous parts in its process, then effectiveness of the clustering result can be indirectly grasped from search performance of the IR system. However, because usually various factors affect overall performance of such application system, it is often difficult to separate contribution of DC from those of other components.

If an 'answer' given by human experts for clustering results is available, then output from clustering algorithms can be evaluated more clearly. Actually, test collections developed mainly

for studying text categorization (e.g, RCV1 [187]) can be also used in DC experiments for the purpose of evaluation. In contrast, when such answer is unavailable, it is necessary to employ an internal criteria for measuring quality of generated clusters.

External evaluation

Suppose that a human expert provides an 'answer', by which a set of 'true' clusters can be constructed in an experiment. The set is denoted by $\mathcal{A} = \{A_1, A_2, \ldots, A_H\}$ here. Clusters generated by an algorithm, $\mathcal{C} = \{C_1, C_2, \ldots, C_L\}$, can be evaluated by using the 'answer' set $\mathcal{A}$. In the case of exclusive text categorization in supervised manner, a topic class of each document in a test set is selected directly from $\{A_1, A_2, \ldots, A_H\}$, and therefore, it is enough to examine whether the assigned class equals to 'true' one or not, and to calculate $v_c = N^{-1} \sum_{i=1}^{N} Ia_i$ where

$$Ia_i = \begin{cases} 1 & \text{if the class assigned to } d_i \text{ equals to its 'true' class} \\ 0 & \text{otherwise} \end{cases} . \tag{3.13}$$

Usually, v_c is called *accuracy* ($0 \leq v_c \leq 1$), which is often used for evaluation of supervised exclusive classification [12].

When results of nonexclusive classification are assessed, recall and precision, or F-measure (Section 2.1.8) are used as evaluation metrics [328], for which it is necessary to count the following numbers of class assignments,

- c_{11}: the number of class assignments to 'true' one
- c_{10}: the total number of class assignments by human experts (i.e., the total number of elements of sets included in $\mathcal{A}$ where a document may belong to two or more sets in $\mathcal{A}$)
- c_{01}: the total number of class assignments by a classification system

and to compute $v_r = c_{11}/c_{10}$ (recall) and $v_p = c_{11}/c_{01}$ (precision). Note that $v_c = v_r = v_p$ in the case of exclusive classification.

Because each document is not directly allocated to a particular answer cluster A_m in usual unsupervised classification, it is necessary for DC to measure the degree of 'closeness' between set $\mathcal{C}$ and set $\mathcal{A}$ (note that it is not always guaranteed that $L = H$). One of the measures is *entropy*,

$$E_k = -\sum_{m=1}^{H} P(A_m|C_k) \log P(A_m|C_k), \tag{3.14}$$

where conditional probability $P(A_m|C_k)$ can be estimated such as

$$\hat{P}(A_m|C_k) = \frac{\tilde{n}_{mk}}{\tilde{n}_k}, \tag{3.15}$$

based on $\tilde{n}_{mk}$ that means the number of documents belonging to both A_m and C_k [293] [13]. When documents in C_k are distributed into many more 'true' clusters, the entropy increases. Thus smaller entropy indicates more successful clustering. In order to evaluate overall a clustering result, it is necessary to compute a weighted average of E_k ($k = 1, \ldots, L$) such as

$$E(\mathcal{A}|\mathcal{C}) = \sum_{k=1}^{L} \frac{\tilde{n}_k}{N} E_k. \tag{3.16}$$

When E_k is replaced by $\max_m(\tilde{n}_{mk}/\tilde{n}_k)$ in Equation (3.16), the indicator is called *purity*.

[12] Actually, class assignments to documents in test collections are checked by *cross validation* approach in the case of supervised classification. If executing 'five-fold' cross validation, then all documents in the test collection are divided into five parts, and the job of class assignments is repeated five times with changing roles of the training set and test set among the five parts (see [84]). Usually, four parts are employed as the training set, and the other part plays a role of the test set in a run.

[13] Equation (3.14) is an 'expectation' of $\log[1/P(A_m|C_k)]$ by probability $P(A_m|C_k)$. See Section 4.1.1 for definition of expectation.

Another well-known measure for gaging closeness between two sets is *mutual information* (MI),

$$M_I(\mathcal{C},\mathcal{A}) = \sum_{m=1}^{H}\sum_{k=1}^{L} P(A_m, C_k)\log\frac{P(A_m, C_k)}{P(A_m)P(C_k)}, \tag{3.17}$$

where $P(A_m)$ and $P(C_k)$ are estimated by $\tilde{n}_m/N$ and $\tilde{n}_k/N$, respectively, and $P(A_m, C_k)$ is $\tilde{n}_{mk}/N$. If $\mathcal{C}$ is completely independent of $\mathcal{A}$, then mutual information $M_I(\mathcal{C},\mathcal{A})$ amounts to zero, which is the minimum [14]. In contrast, the maximum of MI is $\max[E(\mathcal{C}), E(\mathcal{A})]$ where $E(\mathcal{C})$ is entropy of the set $\mathcal{C}$ such that $E(\mathcal{C}) = -\sum_k P(C_k)\log P(C_k)$, and $E(\mathcal{A})$ is defined similarly. Thus *normalized mutual information* (nMI) can be obtained as

$$\tilde{M}_I(\mathcal{C},\mathcal{A}) = \frac{M_I(\mathcal{C},\mathcal{A})}{\max[E(\mathcal{C}), E(\mathcal{A})]}, \tag{3.18}$$

where $0 \le \tilde{M}_I(\mathcal{C},\mathcal{A}) \le 1$ [193]. Another formula for nMI,

$$\breve{M}_I(\mathcal{C},\mathcal{A}) = \frac{M_I(\mathcal{C},\mathcal{A})}{\sqrt{E(\mathcal{C})E(\mathcal{A})}}. \tag{3.19}$$

is also used in literature.

Recall, precision and F-measure can be also considered as a candidate of evaluation metric in unsupervised classification. In the context of DC, when C_k and A_m are given, recall and precision can be defined by $\hat{P}(C_k|A_m) = \tilde{n}_{mk}/\tilde{n}_m$ and $\hat{P}(A_m|C_k) = \tilde{n}_{mk}/\tilde{n}_k$, respectively. Thus F-measure becomes

$$F_m(A_m, C_k) = \frac{2\times\hat{P}(C_k|A_m)\times\hat{P}(A_m|C_k)}{\hat{P}(C_k|A_m)+\hat{P}(A_m|C_k)}, \tag{3.20}$$

which leads to an indicator for evaluating the entire set such as

$$F_s(\mathcal{A},\mathcal{C}) = \sum_{m=1}^{H}\frac{\tilde{n}_m}{N}\max_{k=1,\ldots,L} F_m(A_m, C_k), \tag{3.21}$$

which is called *FScore* or *overall F-measure* [177].

Suppose that a human expert gives an answer of clustering for the sample DB in Table 3.1 such that $A_1 = \{d_1, d_2, d_3\}$, $A_2 = \{d_4, d_5, d_6, d_7\}$ and $A_3 = \{d_8, d_9, d_{10}\}$ (i.e., $H = 3$). Thus the result by k-means clustering when $L = 4$ (see Table 3.4) can be represented as a matrix in Table 3.5, which indicates that members of A_2 and A_3 were divided into two separated clusters, respectively, in the output from the k-means algorithm. Cross tabulation like Table 3.5 is often called *confusion matrix*. For the data, the above indicators are computed as follows:

- Entropy. Since $E_3 = 0.636$ and $E_1 = E_2 = E_4 = 0.0$, $E(\mathcal{A}|\mathcal{C}) = 0.636\times(3/10) = 0.190$.
- Mutual information. Since $M_I(\mathcal{C},\mathcal{A}) = 0.897$, $E(\mathcal{A}) = 1.088$ and $E(\mathcal{C}) = 1.313$, $\tilde{M}_I(\mathcal{C},\mathcal{A}) = 0.683$ and $\breve{M}_I(\mathcal{C},\mathcal{A}) = 0.750$.
- FScore. Since $\max_k F_m(A_1, C_k) = 1$, $\max_k F_m(A_2, C_k) = 0.857$ and $\max_k F_m(A_3, C_k) = 0.666$, $F_s(\mathcal{A},\mathcal{C}) = 0.842$.

If a very poor clustering result was obtained as Table 3.6, then values of these indicators change to $E(\mathcal{A}|\mathcal{C}) = 0.936$, $\tilde{M}_I(\mathcal{C},\mathcal{A}) = 0.111$, $\breve{M}_I(\mathcal{C},\mathcal{A}) = 0.125$ and $F_s(\mathcal{A},\mathcal{C}) = 0.373$.

In a special case that each cluster set obtained from an exclusive DC algorithm can be assigned to an answer set by using a method, it is possible to compute accuracy v_c. Similarly, recall, precision and F-measure based on c_{11}, c_{10} and c_{01} can be employed for nonexclusive DC in such special cases.

[14] If $\tilde{n}_{mk} = 0$, then it is assumed that $P(A_m, C_k) = P(A_m)P(C_k)$.

Table 3.5: Example of confusion matrix (1)

	C_1	C_2	C_3	C_4	Total
A_1	3	0	0	0	3
A_2	0	3	1	0	4
A_3	0	0	2	1	3
Total	3	3	3	1	10

Table 3.6: Example of confusion matrix (2)

	C_1	C_2	C_3	C_4	Total
A_1	1	1	1	0	3
A_2	1	1	1	1	4
A_3	0	1	1	1	3
Total	2	3	3	2	10

Internal evaluation

When any external criterion such as 'true' clusters is not available, *cluster density* is often used as a landmark of success in clustering, and in the case of distance, RSS J in Equation (3.8) can be applied for evaluation of clustering results. For the set of clusters shown in Table 3.4 (i.e., results of k-means clustering), it becomes that $J = 3.035 + 1.172 = 4.207$ when $L = 2$, which is about 34% of total sum of squared errors $\sum_{i=1}^{N} \|\mathbf{d}_i - \mathbf{m}\|^2$ where $\mathbf{m} = N^{-1} \sum_{i=1}^{N} \mathbf{d}_i$ (a centroid vector of the entire set D).

When similarity measure is adopted, an indicator corresponding to the cluster density can be defined as

$$J_S = \sum_{k=1}^{L} \tilde{n}_k \left[\frac{1}{\tilde{n}_k^2} \sum_{i:d_i \in C_k} \sum_{h:d_h \in C_k} s(\mathbf{d}_i, \mathbf{d}_h) \right], \tag{3.22}$$

where $s(\mathbf{d}_i, \mathbf{d}_h)$ means similarity between two vectors (i.e., $s_{ih} = s(\mathbf{d}_i, \mathbf{d}_h)$). Note that J_S is a weighted sum of the means of similarities between all document pairs within each cluster. Otherwise, it is possible to consider the sum of similarities between each document vector and a centroid vector of its cluster,

$$J_{S'} = \sum_{k=1}^{L} \sum_{i:d_i \in C_k} s(\mathbf{d}_i, \mathbf{m}_k). \tag{3.23}$$

Because $\mathbf{m}_k = \tilde{n}_k^{-1} \sum_h \mathbf{d}_h$ for $d_h \in C_k$ (see Equation (3.6)), it becomes that

$$\begin{aligned} \sum_{i:d_i \in C_k} s(\mathbf{d}_i, \mathbf{m}_k) &= \sum_{i:d_i \in C_k} \frac{\mathbf{d}_i^T (\tilde{n}_k^{-1} \sum_h \mathbf{d}_h)}{\|\mathbf{d}_i\| \cdot \|\tilde{n}_k^{-1} \sum_h \mathbf{d}_h\|} \\ &= \frac{1}{\tilde{n}_k} \frac{1}{\|\tilde{n}_k^{-1} \sum_h \mathbf{d}_h\|} \sum_{i:d_i \in C_k} \sum_{h:d_h \in C_k} \frac{\mathbf{d}_i^T \mathbf{d}_h}{\|\mathbf{d}_i\|} \end{aligned} \tag{3.24}$$

by using Equations (3.1). Thus $J_{S'}$ is slightly different from J_S.

For example, if dendrogram by complete linkage in Figure 3.5 is cut at the similarity level of 0.8 (distance level is 0.2), then a set of clusters, $\{d_1\}$, $\{d_2, d_3\}$, $\{d_4\}$, $\{d_5, d_7\}$, $\{d_6, d_8, d_9\}$ and $\{d_{10}\}$, are obtained, and its values of J_S and $J_{S'}$ amount to 9.585 and 9.778, respectively.

3.2 Hierarchical Document Clustering

This section reviews DC based on hierarchical clustering techniques [15]. As mentioned above, dendrogram generated by hierarchical method is useful for IR situations, but unfortunately, its computational complexity is so high. Thus research efforts have been exerted so far for executing hierarchical clustering algorithm on a large set of documents.

3.2.1 Basic algorithm for creating dendrogram

A main reason for high computational complexity of agglomerative hierarchical clustering is that similarities between $O(N^2)$ pairs of documents have to be calculated. If all data to be required for the processing are simply stored into arrays on main memory of a computer, then it is necessary for cosine-based (or unit vector-based) clustering to keep three arrays: (a) array A whose size is $N \times M$ for storing document vectors, (b) array B whose size is N for keeping document norms, and (c) array C whose size is $N(N-1)/2$ for recording values of cosine coefficient (or distance measure). Namely, the size of computer memory for hierarchical DC amounts to $N \times (M+1) + N(N-1)/2$ at least when all data are simply stored in arrays.

Suppose that tf (or tf-idf weight) values and document norms have already been calculated after an indexing process (Section 2.1.6) and stored into arrays A and B. In order to create a dendrogram from the data, the following process is needed:

1. First stage. By checking arrays A and B repeatedly, cosine coefficients are computed and recorded in array C.
2. Second stage. By searching C array repeatedly, clusters or singletons to be sequentially merged are detected.

In the first stage, array A receives $N(N-1) \times M$ accesses in total because fetch of two vectors containing M elements is repeated for $N(N-1)/2$ pairs in computing inner products [16], and array B is checked two times for each of $N(N-1)/2$ pairs in order to know the norms. Therefore, in the first stage, the number of times that values are read out from the arrays, which is denoted by T_A, amounts to

$$\mathrm{T}_A = 2 \times M \times \frac{N(N-1)}{2} + 2 \times \frac{N(N-1)}{2} = N(N-1)(M+1) \tag{3.25}$$

(see Figure 3.13). Of course, other than T_A accesses, arithmetic operations and storage of the results are needed in the first stage.

An example of search algorithm in the second stage was given by Anderberg(1973) [9], which is shown as Figure 3.14 [17]. Similarity matrix calculated in the first stage is stored in array C as a lower triangular matrix like Table 3.3 or Figure 3.15, which is denoted by $\mathrm{C} = [x(i,h)]$ where $i = 2, \ldots, N$ and $h = 1, \ldots, N-1$, and $x(i,h)$ is defined only for $i > h$. Note that $x(i,h) = s_{ih} = s_{hi}$ where $\mathbf{S} = [s_{ih}]$ is the original similarity matrix [18]. Figure 3.14 is an algorithm of single linkage clustering, but complete linkage clustering can be executed by defining as $h'(i) = \arg\min_h x(i,h)$ in step 1) and by replacing 'max' by 'min' in step 3-1).

An illustrative description of the Anderberg's algorithm is shown as Figure 3.15, in which d_3 and d_5 are first merged since $i' = 5$ and, $h'(i') = 3$ (i.e., $x(5,3) = 0.50$, which is the maximum value). According to a rule in step 3-1) in Figure 3.14, data of the merged cluster will be recorded into i'-th position, and therefore, the value of $x(3,2)$ in 3rd row is moved to $x(5,2)$ because $s_{32} > s_{52}$. In the case that $s_{34} > s_{54}$, the value of $x(4,3)$ in 4th row (not 3rd row) has to be copied into $x(5,4)$ because C is a lower triangle matrix, which makes actual source code slightly complicated.

Computational complexity of the Anderberg's algorithm can be analyzed as follows [9]:

[15] The review is partly based on Kishida(2003) [147]. There are other review articles covering contents of this section, next section and next chapter (e.g., Berkhin(2002) [22], Andrew & Fox(2007) [12], Jain(2010) [135] and Aggarwal & Zhai(2012) [4]).

[16] Actually, it is not necessary to keep M values straightforwardly in the array since real document vectors are very sparse. More efficient way for DC will be discussed later.

[17] In other parts of this thesis, s is used for indicating number of a step in iterative computation, not b.

[18] In the case of distance-based single linkage, 'min' operator is used instead of 'max'.

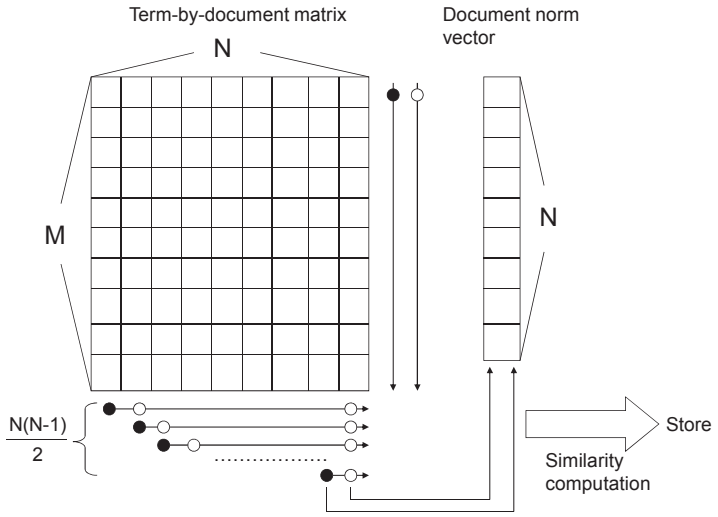


Figure 3.13: Naive computation of similarity matrix

‡ Anderberg's algorithm for detecting clusters to be merged (single linkage)

Set: C $= [x(i,h)]$ and $b = 1$, where C is a lower triangular matrix of similarities $\mathbf{S} = [s_{ih}]$ $(i, h = 1, \ldots, N)$, namely, $x(i,h) = s_{ih} = s_{hi}$ and $i > h$, and b indicates number of a step in iteration.

1) Initially, find the maximum score $\max_h x(i,h)$ by each row $(i = 2, \ldots, N)$, and store $h'(i) = \arg\max_h x(i,h)$.

2) Find $i' = \arg\max_i x(i, h'(i))$ where $i = 2, \ldots, N - b + 1$.

3) Merge i'-th data and h'-th data $(h' \equiv h'(i'))$ and update the matrix.

   3-1) Update scores of i'-th row such that $x(i', k) = \max(s_{i'k}, s_{h'k})$ (when $i' > k$) or $x(k, i') = \max(s_{i'k}, s_{h'k})$ (when $i' < k$), where $k = 1, \ldots, N - b + 1$ and $k \neq i', h'$. In this process, h'-th object is added to the cluster corresponding to i'-th row as its member.

   3-2) Remove h'-th row and column from the matrix, and re-assign index i to remaining $N - b - 1$ rows and columns for keeping a lower triangular matrix. Also, $h'(i')$ is updated because h'-th data are removed.

4) Update b such that $b \leftarrow b + 1$. If $b < N$, then return to step 2). Otherwise, the procedure is terminated.

Out: A dendrogram.

Figure 3.14: Anderberg's single linkage algorithm

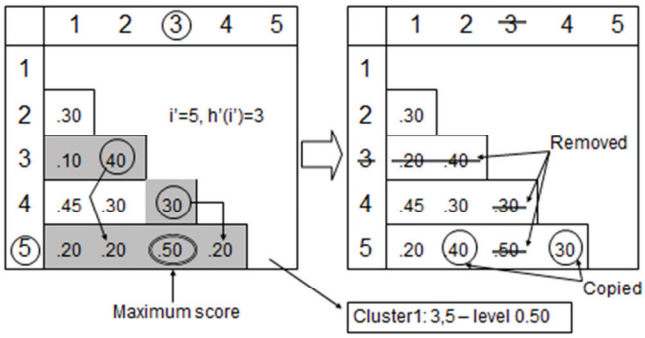


Figure 3.15: Outline of Anderberg's single linkage algorithm

- Step 1) At i-th row, $i-2$ comparisons are needed for detecting its maximum value, and therefore, it is necessary to compare values $\sum_{i=3}^{N}(i-2) = 2^{-1}(N-1)(N-2)$ times in total from third to N-th rows [19].
- Step 2) In each iteration, $N-b$ rows remain in the matrix and $N-b-1$ comparisons has to be done for detecting i' $(= \arg\max_i x(i, h'(i)))$. Thus because Step 2) is repeated $N-1$ times, the total number of comparisons amounts to $\sum_{b=1}^{N-1}(N-b-1) = 2^{-1}(N-1)(N-2)$.
- Step 3-1) Since there are $N-b+1$ clusters in b-th iteration (note that a singleton is also counted as a cluster), $N-b-1$ $(= (N-b+1)-2)$ elements of i'-th cluster have to be compared with those of h'-th cluster. Thus the total number of comparisons amounts to $\sum_{b=1}^{N-1}(N-b-1) = 2^{-1}(N-1)(N-2)$.
- Step 3-2) Because the number of elements in a new i'-th cluster is $N-b-1$, its elements have to be compared $N-b-2$ times for detecting new $h'(i')$. Thus the total number of comparisons amounts to $\sum_{b=1}^{N-2}(N-b-2) = 2^{-1}(N-2)(N-3)$.

Therefore, the total number of comparisons in Anderberg's algorithm can be obtained such that

$$\mathrm{T}_C = 3 \times \frac{1}{2}(N-1)(N-2) + \frac{1}{2}(N-2)(N-3) = 2N^2 - 7N + 6 \tag{3.26}$$

by summing up those in Step 1) to 3) [20]. Note that Equation (3.25) indicates the number of 'accesses' for reading out values from main memory whereas Equation (3.26) provides the number of 'comparisons' for which values have to be extracted from the memory. Therefore, T_C can not be compared directly with T_A.

Although it is generally difficult to assess exactly computational complexity, it turns out that basic algorithm of creating a dendrogram needs at least memory size of $N \times (M+1) + N(N-1)/2$, $N(N-1)(M+1)$ checks of the memory for creating similarity matrix, and $2N^2-7N+6$ comparisons for finding clusters to be merged (of course, other additional areas or operations are needed). Table 3.7 shows actual values of them in three cases that $N = 1000$, $N = 10000$ and $N = 100000$ under an assumption that $N = M$. Even though M can be reduced by feature selection (Section 3.1.3), it would be easy to understand difficulty of processing a large number of documents containing many terms in agglomerative hierarchical clustering from Table 3.7.

3.2.2 Hierarchical clustering for large-scale document set

Use of index file

As mentioned above, a term-by-document matrix like that in Figure 3.13 is generally sparse, and it is possible that there are many document pairs sharing no term. Because the cosine coefficient of

[19] The sum of an arithmetic sequence $a_1, \ldots, a_n$ is calculated as $(1/2)[n(a_1+a_n)]$. In this case, $a_1 = 1$, $n = N-2$ and $a_n = N-2$.

[20] The number is not equivalent to that by Anderberg(1973) [9] because different operations in Step 3) are assumed in Figure 3.14.

Table 3.7: Example of computational complexity (when $N = M$)

N	$N(M+1)$	$N(N-1)/2$	$N(N-1)(M+1)$	$2N^2 - 7N + 6$
10^3	1,001,000	499,500	999,999,000	1,993,006
10^4	100,010,000	49,995,000	999,999,990,000	199,930,006
10^5	10,000,100,000	4,999,950,000	999,999,999,900,000	19,999,300,006

such pairs becomes automatically zero, they can be skipped in the process of computing a similarity matrix, and if so, the number of pairs to be processed becomes smaller than $N(N-1)/2$.

When an index file was created in an indexing process, it is easy to limit computation of the cosine coefficient to only document pairs sharing one or more terms by using the index file [318]. For example, Figure 3.16 shows an index file of the sample DB in Table 3.1. An entry of the file includes an index term, document frequency n_j and document identifiers (IDs) containing the term with frequencies f_{ij}. The file is implemented so that each entry term can be accessed very quickly by using a search algorithm such as hashing, binary search, B-tree and so on (When the document set is large, the B-tree would be appropriate as discussed later).

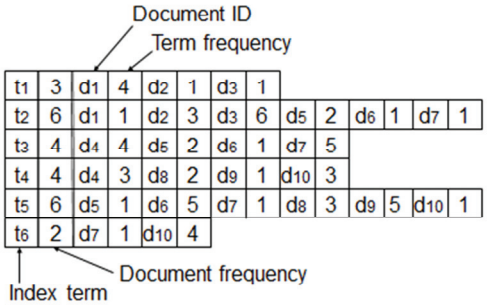


Figure 3.16: Example of index file (for the sample DB)

By checking the index file from its top position sequentially, it is feasible to know that document d_1 shares terms with only d_2, d_3, d_5, d_6 and d_7, and that the cosine coefficient with the other documents becomes automatically zero. Thus it is expected that complexity of computing the cosine coefficient reduces by using the index file.

Another efficient hierarchical DC technique is to repeat hierarchical clustering M times by each index term, which creates M dendrograms [70]. For example, for t_1 in Figure 3.16, a dendrogram including d_1, d_2, and d_3 is constructed, and by repeating it for $t_2, \ldots, t_6$, six dendrograms are obtained from the index file. Since document pairs not sharing any term are automatically removed in the process of computing similarity, its complexity would decrease if the numbers of documents recorded for each index term are enough small (otherwise, it is possible to increase computational complexity [112]). Of course, this technique is applicable to only a special purpose such as cluster-based IR (Chapter 1) because it produces separate M dendrograms, and it would be hard to regard them as a partition of the entire set $C_1, \ldots, C_L$.

Implementation of single linkage method for large document set

In this section, suppose that a document set is very large, and that areas of $N \times (M+1)$ (Figure 3.13) and of $N(N-1)/2$ (Figure 3.15) can not be allocated in main memory. Also, for such document set, it is usually impossible to place its index file straightforwardly on the memory. When all data for clustering can not be stored in main memory, a larger device such as hard disk (HDD) has to be employed [9] even though access to this sort of devices is very slow than to main memory.

First, suppose that document vectors obtained by an indexing process are stored in a file on a

HDD shown as Figure 3.17, which is called 'document vector file' for convenience. In the document vector file, records consisting of document ID, index terms and term frequencies are stored and accessible sequentially.

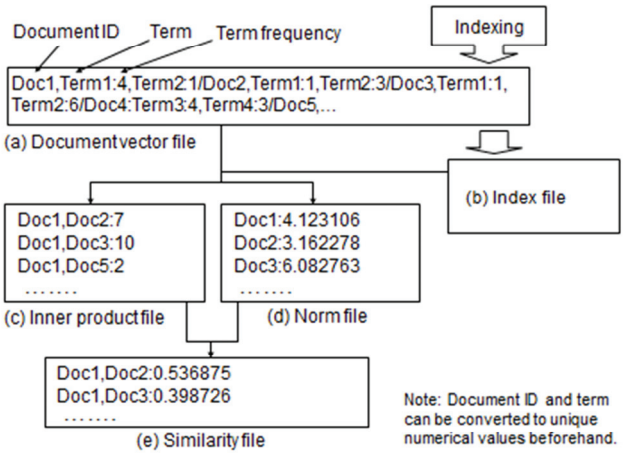


Figure 3.17: Framework of hierarchical document clustering using HDD files

By reading out the record in the document vector file and checking the index file repeatedly for each term included in the record, it is feasible to compute a) inner products with other documents sharing one or more terms and b) norm of the document. For example, the first record of the document file is "Doc1,Term1:4,Term2:1", and information on Term1 and Term2,

- Term1: $n_1 = 3$, $f_{11} = 4$, $f_{21} = 1$ and $f_{31} = 1$,
- Term2: $n_2 = 6$, $f_{12} = 1$, $f_{22} = 3$, $f_{32} = 6$ $f_{52} = 2$, $f_{62} = 1$ and $f_{72} = 1$,

can be obtained by searching the index file (see Figure 3.16). Thus each inner product between d_1 and other documents can be computed because the above list includes all terms whose tf is not zero in d_1, and terms not contained in the list have no contribution to the degree of similarity with d_1. Actually, in the case of adopting tf vector, inner products are computed such that

$$\sum_{j=1}^{M} w_{1j} w_{2j} = 4 \times 1 + 1 \times 3 = 7,$$

$\sum_j w_{1j} w_{3j} = 10$, $\sum_j w_{1j} w_{5j} = 2$, $\sum_j w_{1j} w_{6j} = 1$ and $\sum_j w_{1j} w_{7j} = 1$. Of course, tf-idf weights can be calculated because document frequencies ($n_1 = 3$ and $n_2 = 6$) are known from the index file concurrently [21].

Unfortunately, because vector norm of documents other than that of d_1 are unknown, the cosine coefficient can not be computed when the record of d_1 was checked. Therefore, values of the inner product have to be temporarily stored into a file on HDD, which is called 'inner product file' for convenience (see Figure 3.17). Since only the norm of d_1 can be computed in this time, its value is concurrently stored into another file on HDD, which is called 'norm file' here (see Figure 3.17). Otherwise, it may be possible to keep norm information in main memory since size of the norm file is just $O(N)$.

By using the inner product file and norm file, a 'similarity file' recording values of the cosine coefficient between all pairs can be created afterward (see Figure 3.17). More specifically, a record of the inner product file is first read out, and then the norm file is searched to extract norm values of the two documents, which allows to compute the cosine coefficient between them. In the case that the norm information is stored on the main memory, this operation would be very easy. When the norm file is on HDD, the B-tree may be appropriate as an algorithm for the searches (see below).

[21] An experiment on the algorithm based on tf-idf weighting was attempted by Kishida(2002) [146].

In order to construct a dendrogram from the similarity file, records in it have to be sorted in descending order of similarity scores [9]. For example, in the case of the sample DB, top-ranked pairs in the similarity file are as follows:

Doc2, Doc3, 0.987763
Doc6, Doc9, 0.943564
Doc8, Doc9, 0.924678
.....

In the case of single linkage clustering, it is easy to find the first cluster $\{d_2, d_3\}$ and the second cluster $\{d_6, d_9\}$ by reading the file from its top position. Furthermore, by recoding a unique number of current cluster to which each document belongs, document d_8 can be marged into the second cluster when examining the third record "Doc8, Doc9, 0.924678" [22]. In the procedure, records of two documents which have been already allocated to an identical cluster are skipped. After all records in the similarity file are processed, a complete dendrogram by the single linkage will be obtained.

A practical advantage of the single linkage method is that clusters to be merged in the process of creating a dendrogram can be identified only by 'linearly' checking the similarity file from the top to the bottom if records in the file are sorted by their similarity scores. Therefore, the similarity file can be on a HDD, which allows to execute the algorithm for a large document set not stored in main memory. In such situation, it is necessary to use HDD files for implementation of the index file and for execution of sorting the similarity file, which will be described next.

B-tree and merge-sort

Figure 3.18 shows an illustrative description of the *B-tree* [280]. Whereas all data of an index file indicated in Figure 3.16 are stored in a file on HDD at right-hand side of Figure 3.18, main memory has only a subset of index terms. Thus a huge index file can be implemented even if size of the memory is small (the number of index terms registered on the memory can be reduced arbitrarily at the cost of access speed). The subset of terms is stored into a balanced tree on the memory as shown in Figure 3.18.

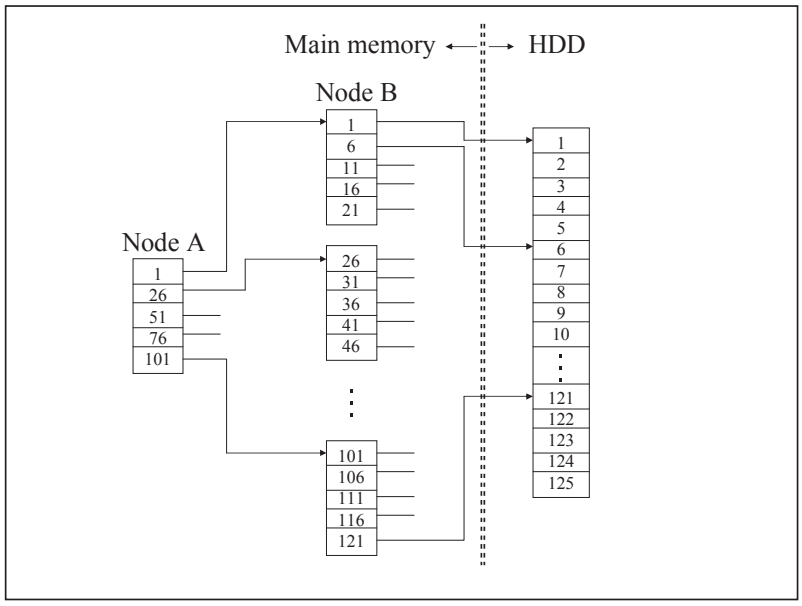


Figure 3.18: Example of B-tree

Suppose that each index term is represented by a unique number ("1" $\sim$ "125") after string

[22]It is feasible to detect quickly information that $d_9 \in C_2$ if an array whose size is $O(N)$ can be allocated in the main memory for storing a cluster number to which each document is belonging. Of course, this array has to be frequently rewritten when two clusters are merged in the process of creating a dendrogram.

sorting as in Figure 3.18, and that the tree is searched for number "8". First, cells of "Node A", "1", "26", "51", "76" and "101", are checked sequentially. Since "1" < "8", "26" in next cell is again compared with the search key "8". As a result, it turns out that the value "26" is bigger than "8". In this case, the process moves back to the previous cell "1" and goes to next node "B" by using a pointer incorporated in the cell "1". By repeating the same procedure on node "B", cell "6" is identified. After moving to the position of "6" in a file on HDD due to the fact that node "B" is located at the bottom level of the tree, data of "8" can be found by reading records on HDD sequentially from that of index term "6". Note that any information of "8" does not exist on the main memory.

When a B-tree is constructed, a) size of the subset of terms registered in the tree and b) the number of terms placed on each node, can be adjusted depending on the situation. In this example, five terms are stored in each node, and terms in the nodes are selected every five ones (i.e., 1st, 6th, 11th, ...) from the set of all index terms. If the number of terms stored in each node increases, then hierarchical levels of the tree on memory reduces (which may lead to accelerating the process) but more processing time on each node may be required. Also, when the number of terms selected for the nodes decreases (e.g., every ten index terms), size of the memory storing the nodes becomes smaller but more processing time on HDD would be needed.

It may be not easy to construct a large-scale index file because all data for it can not enter into main memory. In this case, after dividing appropriately the target document file into some subfiles so as to allow a single subfile to be processed on the memory, data for index terms, document frequencies and term frequencies have to be generated by each subfile, respectively. Final index file can be created by merging the data from the subfiles, which is feasible by using a *merge-sort* algorithm (e.g., see [280] for details). Although more processing time is inevitably needed in the case of dividing the target file into small subfiles, a large-scale index file can be created by not sizable memory. The combination of merge-sort and B-tree is a very useful tool for clustering large document sets on relatively small memory [23].

3.2.3 Voorhees' algorithm

Single linkage algorithm by Voorhees

Algorithms proposed by Voorhees(1986) [309] try to execute hierarchical clustering by using only $O(N)$ space of main memory, not $O(N^2)$ as Figure 3.15. For instance, in the case of single linkage clustering, four arrays with size N, $nn[\,]$, $sim[\,]$, $InHier[\,]$ and $sim2[\,]$, are mainly employed in the process of constructing a dendrogram. When this algorithm is applied to the sample DB in Table 3.1, d_1 is first selected, and similarity measures between d_1 and other $N-1$ documents are computed using an index file [308]. The result is recorded in $sim2[\,]$ (i.e., $sim2[2] \leftarrow 0.54$, $sim2[3] \leftarrow 0.40$, ..., $sim2[10] \leftarrow 0.00$). In i-th cell of $nn[\,]$, an index number of the document having the maximum similarity with d_i is kept (e.g., $nn[2] = 1$, which means that d_1 is the most similar with d_2 in the current stage). Concurrently, its similarity score is stored in i-th cell of $sim[\,]$ (e.g., $sim[2] = 0.54$). The fact that $nn[2] = 1$ and $sim[2] = 0.54$ can be found by checking sequentially from $sim2[2]$ to $sim2[N]$. As a result, d_1 and d_2 are incorporated into the dendrogram, which is recorded in $InHier[\,]$ such as $InHier[2] \leftarrow 1$ (since a document selected first, i.e., d_1, is always added into the dendrogram, it is not necessary to record it).

In next stage, d_2 becomes the target document instead of d_1, which is recorded in variable $CurrId$ such as $CurrId \leftarrow 2$. By searching again the index file, similarity measures between d_2 and other documents are computed and recorded in $sim2[\,]$, from which $nn[\,]$ and $sim[\,]$ keeping information on nearest neighbors of each document may be updated. Since d_3 has the maximum score of similarity in $sim[\,]$ (with d_2) among documents not yet entering into the dendrogram, d_3 and its nearest neighbor d_2 are inserted into the dendrogram and $InHier[3] \leftarrow 1$. This procedure is repeated until all documents are in the dendrogram.

More specifically, the detail of Voorhees' algorithm for single linkage is described in Figure 3.19, and Table 3.8 shows states of $nn[\,]$ and $sim[\,]$ in each step when the algorithm is applied to the sample DB. After updating $nn[\,]$ and $sim[\,]$ in step 2-1), the algorithm tries to find the maximum score from cells in $sim[\,]$ under a condition that $InHier[i] = 0$ in step 2-2). In step 3), a tuple of

[23] In experiments reported in Chapter 5, this system was used.

a) the document having the maximum similarity ($NextID$), b) its nearest neighbor ($nn[NextID]$) and c) its similarity score ($MaxSim$), is inserted into the dendrogram. For example, in Table 3.8, when $CurrID = 3$, it turns out that $NextID = 5$ and $nn[NextID] = 3$ (its similarity score is 0.66) before moving to step 3). Therefore, a tuple of 5, 3 and 0.66 is added to the dendrogram, which is used for developing the dendrogram.

‡ Voorhees' algorthm of single linkage clustering

Set: For $i = 2, \ldots, N$, $nn[i] \leftarrow 0$, $InHier[i] \leftarrow 0$ and $sim[i] \leftarrow 0.0$. $CurrID \leftarrow 1$.

1) $InHier[CurrID] \leftarrow 1$ and similarity scores between the document of $CurrID$ and other documents are computed and recorded in $sim2[\,]$ (from $sim2[2]$ to $sim2[N]$).

2) $MaxSim \leftarrow 0.0$, $NextId \leftarrow 0$ and $i \leftarrow 2$.

   2-1) In the case that $InHier[i] = 0$, if $sim2[i] > sim[i]$ then $sim[i] \leftarrow sim2[i]$ and $nn[i] \leftarrow CurrId$.

   2-2) In the case that $InHier[i] = 0$, if $sim[i] > MaxSim$ then $MaxSim \leftarrow sim[i]$ and $NextId \leftarrow i$.

   2-3) $i \leftarrow i + 1$. If $i > N$ then go to next step 3). Otherwise, return to step 2).

3) If $NextId = 0$ then terminate the procedure. Otherwise, insert $NextID$, $nn[NextID]$ and $MaxSim$ into the dendrogram, and $CurrID \leftarrow NextID$ and return to step 1).

Out: A dendrogram

Figure 3.19: Voorhees' single linkage algorithm

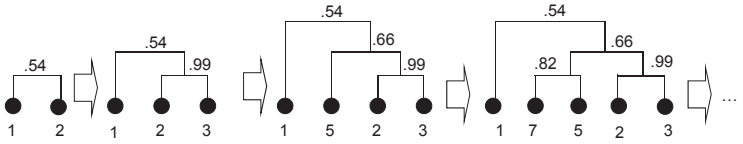


Figure 3.20: Development of dendrogram in Voorhees' algorithm

The most-right column of Table 3.8 indicates all information to be inserted into the dendrogram, from which a hierarchy can be constructed as shown in Figure 3.20. If development of the hierarchy is continued, then a dendrogram shown in Figure 3.5 (a) is finally obtained. Since creating a new cluster does not affect similarities between other clusters in single linkage clustering, it is possible to detect straightforwardly a pair of documents defining a similarity between clusters. For example, when $CurrID = 4$ in Table 3.8, a pair of d_5 and d_6 connecting two large clusters $\{d_2, d_3, d_4, d_5, d_7\}$ and $\{d_6, d_8, d_9\}$ is extracted smoothly by keeping the nearest neighbor of d_6.

In step 1) of the algorithm, similarity scores between a document recorded in $CurrID$ and other documents have to be computed, and therefore, it is not possible to avoid computations for $O(N^2)$ pairs in total. Practically, document vector file, index file, norm file shown in Figure 3.17 can be used for the computation.

Complete linkage algorithm by Voorhees

In the case of complete linkage clustering, the algorithm has to find a 'maximum' link between clusters, which is defined as a 'minimum' similarity between a document of the one cluster and a document of the other cluster. Because the 'maximum' and 'minimum' are mixed in it, the algorithm is more complicated than that of single linkage clustering. In the algorithm of Voorhees

Table 3.8: Execution of Voorhees' single linkage algorithm

	$nn[]$ (top) and $sim[]$ (bottom)									
$CurrID$	2	3	4	5	6	7	8	9	10	Insert
1	1	1	0	1	1	1	0	0	0	2,1
	.54	.40	.00	.16	.05	.05	.00	.00	.00	.54
2	*	2	0	2	2	2	0	0	0	3,2
	-	.99	.00	.63	.18	.18	.00	.00	.00	.99
3	*	*	0	3	3	3	0	0	0	5,3
	-	-	.00	.66	.19	.19	.00	.00	.00	.66
5	*	*	5	*	5	5	5	5	5	7,5
	-	-	.53	-	.58	.82	.28	.33	.07	.82
7	*	*	7	*	5	*	5	5	7	4,7
	-	-	.76	-	.58	-	.28	.33	.19	.76
4	*	*	*	*	5	*	4	5	4	6,5
	-	-	-	-	.58	-	.33	.33	.35	.58
6	*	*	*	*	*	*	6	6	4	9,6
	-	-	-	-	-	-	.80	.94	.35	.94
9	*	*	*	*	*	*	9	*	4	8,9
	-	-	-	-	-	-	.92	-	.35	.92
8	*	*	*	*	*	*	*	*	8	10,8
	-	-	-	-	-	-	-	-	.49	.49

Note: * means that $InHier[i] = 1$.

(1986) [309], all document pairs are first sorted in descending order of similarities, and checked from the top to the bottom for finding the 'maximum' similarly to single linkage clustering. However, unlike single linkage clustering, the number of document pairs between specific two clusters is concurrently counted during the checking.

For example, suppose that there are four documents d_A, d_B, d_C and d_D, and that a sorted list in descending order of similarity such as "1.(d_A, d_B), 2.(d_C, d_D), 3.(d_A, d_C), 4.(d_A, d_D), 5.(d_B, d_C) and 6.(d_B, d_D)" can be obtained. Since (d_A, d_B) and (d_C, d_D) appear in top of the list, clusters $\{d_A, d_B\}$ and $\{d_C, d_D\}$ are generated at first and second checks of the list, respectively. The number of document pairs between the two clusters is $2 \times 2 = 4$. Thus when 'fourth' pair (d_B, d_D) among four document pairs between the two clusters is checked lastly, it turns out that similarity of (d_B, d_D) is the minimum among the four pairs. This means that the pair corresponds to the complete linkage between $\{d_A, d_B\}$ and $\{d_C, d_D\}$ and that the two clusters are merged into a cluster at the level of similarity between pair (d_B, d_D). As this example shows, in order to find complete linkage between clusters, it is necessary to record link information containing IDs of the linked cluster and the number of document pairs already checked in the sorted list. In the case of large-scale DC, it is expected that large space is needed for keeping the information.

Actually, when a new cluster is created or a singleton is appearing, it is necessary to update carefully link information related to the new cluster. Suppose that there is another sorted list such as

1.(d_A, d_B), 2.(d_A, d_C), 3.(d_C, d_E), 4.(d_C, d_D), 5.(d_A, d_D), 6.(d_B, d_E),
7.(d_B, d_C), 8.(d_B, d_D), 9.(d_C, d_E), 10.(d_D, d_E)

including five documents. First cluster generated by reading "1.(d_A, d_B)" is denoted by "C1[A,B]" for convenience where "1" in "C1" indicates that the cluster ID is 1. When "2.(d_A, d_C)" is checked, a singleton "C2[C]" has to be created, and link information is recorded as "C1[A,B]-L[2,1]" and "C2[C]-L[1,1]" where the first argument of L[ ] is ID of a linked cluster and the second indicates the cumulative number of document pairs to be checked. The cumulative number tells us whether the checked document pair is corresponding to a complete linkage or not. In this example, a dendrogram is developed as follows:

1.$(d_A, d_B) \rightarrow$ C1[A,B]

2.$(d_A, d_C) \rightarrow$ C1[A,B]-L[2,1], C2[C]-L[1,1]
3.$(d_A, d_E) \rightarrow$ C1[A,B]-L[2,1]-L[3,1], C2[C]-L[1,1], C3[E]-L[1,1]
4.$(d_C, d_D) \rightarrow$ C1[A,B]-L[2,1]-L[3,1], C2[C,D]-L[1,1], C3[E]-L[1,1]
5.$(d_A, d_D) \rightarrow$ C1[A,B]-L[2,2]-L[3,1], C2[C,D]-L[1,2], C3[E]-L[1,1]
6.$(d_B, d_E) \rightarrow$ C1[A,B,E]-L[2,2], C2[C,D]-L[1,2]
7.$(d_B, d_C) \rightarrow$ C1[A,B,E]-L[2,3], C2[C,D]-L[1,3]
8.$(d_B, d_D) \rightarrow$ C1[A,B,E]-L[2,4], C2[C,D]-L[1,4]
9.$(d_C, d_E) \rightarrow$ C1[A,B,E]-L[2,5], C2[C,D]-L[1,5]
10.$(d_D, d_E) \rightarrow$ C1[A,B,C,D,E]

When "6.(d_B, d_E)" is checked, L[3,1] of C1[A,B] becomes L[3,2], which leads to merging of C1[A,B] and C3[E] because the total number of pairs between the two clusters is $2 \times 1 = 2$. Similar operation is repeated when reading lastly "10.(d_D, d_E)".

As this example shows, for executing complete linkage clustering, enough space on main memory is needed for storing information of created clusters (e.g., C1[A,B]) and of links between clusters (e.g., L[2,1]) [24]. In the case that linkage between clusters spreads widely, large space may be required. Needless to say, similarity computation and sorting operation are also indispensable.

3.2.4 Binary divisive clustering

Bisecting k-means clustering

Agglomerative hierarchical clustering of documents discussed so far can be characterized as a bottom-up approach starting with amalgamations of individual documents, in which inevitably similarities of $O(N^2)$ pairs has to be computed. In contrast, *divisive hierarchical clustering*, which is a top-down approach, can construct a hierarchy with no similarity (or distance) matrix. Namely, the entire document set D is first divided into two nonempty parts, and each part is recursively partitioned in the same way, by which a hierarchy is finally obtained.

The algorithm is also computationally expensive because it is necessary to consider $2^{N-1} - 1$ possible divisions of the target data in the first stage [322]. However, if the data in each stage is approximately divided in two parts without precisely considering all possible divisions, then its computational complexity may reduce. For instance, in *bisecting k-means* clustering [293, 337], the entire document set D is first partitioned into two parts by a k-means algorithm (e.g., Figure 3.8). Figure 3.21 shows a hierarchy of the sample DB, which was obtained based on the bisecting k-means algorithm by using results from kmeans( ) function of R [25] (note that the k-means clustering is based on distance between unit vectors). Namely, in Figure 3.21, k-means clustering with $L = 2$ and $r' = 10$ is repeated six times (where L indicates the number of clusters as before and r' denotes the maximum number of iterations), first partitioning operation of which divided D into $\{d_1, d_2, d_3, d_5, d_6, d_8, d_9\}$ and $\{d_4, d_7, d_{10}\}$. In next k-means clustering, the former cluster was split into $\{d_1, d_2, d_3, d_5\}$ and $\{d_6, d_8, d_9\}$. Since clusters consisting of only two documents are automatically divided without any effort, the total times of executing the k-means algorithm (i.e., kmeans( )) amounts to six. In this example, the resulting hierarchy is almost same with those by complete linkage clustering in Figure 3.5(b) and the Ward's method in Figure 3.7(a) except for the location of d_5.

Suppose that the resulting hierarchy becomes a completely 'balanced' tree. For example, it is assumed that a set including 16 documents ($N = 16$) was divided into two subsets of equal size with 8 documents, and similarly, each subset was partitioned into two subsets with 4 documents. If they were finally decomposed into subsets with 2 documents, respectively, then the total number of documents scanned for all executions of k-means clustering amounts to $16 + (8 \times 2) + (4 \times 4) = 16 \times 3 = 48$ under an assumption that $r' = 1$. In general, this number can be computed such that

$$\sum_{h=0}^{\log_2 N - 2} 2^h \times \frac{N}{2^h} = \frac{1}{2}(\log_2 N - 2 + 1)(N + N) = N(\log_2 N - 1), \tag{3.27}$$

[24] See Voorhees(1986) [309] for more specific implementation of the algorithm for complete linkage clustering. This paper also discussed an algorithm for group average linkage.

[25] Actually, the Hartigan-Wong algorithm [119] (see Section 3.3.1) was used as a method of k-means clustering.

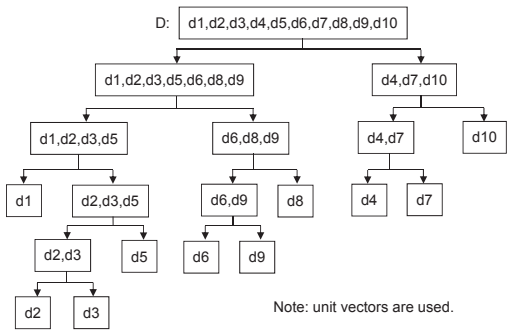


Figure 3.21: Example of bisecting k-means clustering (based on results by R-2.12.0)

Table 3.9: Number of comparisons $2N(\log_2 N - 1)r'$ in bisecting k-means

		Maximum iterations r'		
N	$N(N-1)/2$	1	5	10
2^{10}=1,024	523,776	18,432	92,160	184,320
	100%	3.52%	17.60%	35.19%
2^{14}=16,384	134,209,536	425,984	2,129,920	4,259,840
	100%	0.32%	1.59%	3.17%
2^{18}= 262,144	34,359,607,296	8,912,896	44,564,480	89,128,960
	100%	0.03%	0.13%	0.26%
2^{20}= 1,048,576	549,755,289,600	39,845,888	199,229,440	398,458,880
	100%	0.01%	0.04%	0.07%

Note: A completely balanced tree is assumed.

which means that $\mathcal{T}(N) = 2 \times N(\log_2 N - 1) \times r'$ comparisons between documents and cluster centroids has to be required at the maximum for execution of the bisecting k-means clustering (note that always $L = 2$). Table 3.9 shows examples of the number of comparisons in the bisecting k-means algorithm with percentages of $2N(\log_2 N - 1)r'$ for $N(N-1)/2$, which can be computed as

$$\mathcal{P}_B(N) = \frac{2N(\log_2 N - 1)r'}{N(N-1)/2} = \frac{4(\log_2 N - 1)r'}{N-1} \tag{3.28}$$

(note that cluster generation process is also needed after computing $N(N-1)/2$ similarities in agglomerative hierarchical clustering).

As Table 3.9 and Equation (3.28) indicate, it is expected that advantage of the bisecting k-means clustering on computational complexity rapidly increases as the document set becomes larger, in comparison with agglomerative hierarchical clustering, under an assumption that a balanced tree is obtained by the bisecting k-means clustering. If this assumption is not valid, then the situation becomes worse. For example, the number of comparisons in Figure 3.21 is actually 600 at maximum, which is greater than $2 \times 10 \times (\log_2 10 - 1) \times 10 \simeq 464.3$. In an unbalanced tree shown in Figure 3.6(a), the total number of comparisons between documents and clusters becomes

$$\mathcal{T}(N) = 2 \times \sum_{h=0}^{N-3} N - h = (N-2)(N+3) = N(N+1) - 6, \tag{3.29}$$

which is greater than $N(N-1)/2$ [26]. Therefore, whether the bisecting k-means clustering is actually efficient or not is dependent on the degree of balance in the resulting hierarchy.

Although the partitioning operation of a document subset was repeated until all the subsets reduce to a single document in Figure 3.21, it may not be necessary to split a subset any more

[26] When $D = \{d_A, d_B, d_C, d_D, d_E\}$ (i.e., $N = 5$), an unbalanced tree is generated such as $\{d_A, d_B, d_C, d_D, d_E\} \rightarrow \{d_A\} \cup \{d_B, d_C, d_D, d_E\}$, $\{d_B, d_C, d_D, d_E\} \rightarrow \{d_B\} \cup \{d_C, d_D, d_E\}$, and $\{d_C, d_D, d_E\} \rightarrow \{d_C\} \cup \{d_D, d_E\}$. Thus the total number of comparisons amounts to $10 + 8 + 6 = 24 = 5 \times 6 - 6$ if $r' = 1$.

if the subset is enough homogeneous. Namely, a subset to which the k-means algorithm is again applied can be selected at each stage according to a criterion such as RSS in Equation(3.5). For instance, it is feasible to divide a given subset only if the criterion is improved by the division [337].

Also, the PDDP algorithm [31, 32, 164] which splits repeatedly document sets into two separate parts based on results of principal component analysis can be also categorized into the divisive hierarchal clustering. This method will be discussed in Section 4.2.

Combination of divisive and agglomerative methods

It is possible to combine divisive and agglomerative approaches. For instance, in *constrained agglomerative clustering* [337, 339], k-means clustering is executed for the entire set D at first stage, and at second stage, dendrograms are generated from each resulting subset, respectively (see Figure 3.22). Suppose that D is partitioned into L groups by a k-means algorithm at first stage and the groups contain $\tilde{n}_1, ..., \tilde{n}_L$ documents, respectively. Thus it is expected that

$$O(N^2) > \sum_{k=1}^{L} O(\tilde{n}_k^2), \tag{3.30}$$

about the number of document pairs. Also, if the k-means algorithm identifies homogeneous groups successfully, then quality of dendrograms created at second stage may be higher than that in applying a hierarchical clustering algorithm directly to the entire set D. If needed, the k-means algorithm can be repeatedly applied at first stage in a manner of the binary divisive clustering, or it is fe

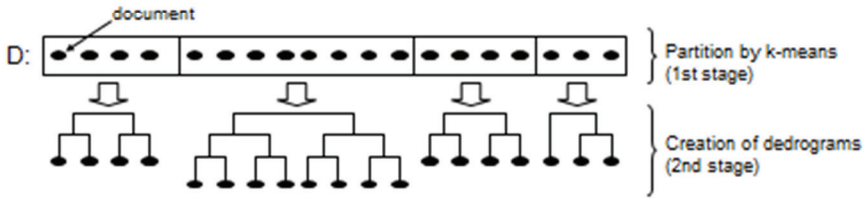


Figure 3.22: Outline of constrained agglomerative clustering

For simplifying mathematical analysis, it is assumed that all groups created at first stage of the constrained agglomerative clustering include an equal number of documents $\tilde{n}$ (i.e, $\tilde{n} = N/L$). In this case, the number of comparisons between documents and clusters or between document pairs becomes that

$$\begin{aligned} \mathcal{T}(N) &= NLr' + L \times \tilde{n}(\tilde{n}-1)/2 = NLr' + L(N/L)(N/L-1)/2 \\ &= NLr' + \frac{N^2}{2L} - \frac{N}{2}. \end{aligned} \tag{3.31}$$

Since $d\mathcal{T}(N)/dL = Nr' - N^2/(2L^2)$ [27], by solving an equation $d\mathcal{T}/dL = 0$, an optimum number of L providing the smallest number of comparisons can be obtained such that $\tilde{L} = \sqrt{N/(2r')}$.

Figure 3.23 shows the number of comparisons $\mathcal{T}(N)$ in Equation (3.31) when $N = 5000$, $N = 8000$, $N = 10000$ and $r' = 1$. In the case that $N = 5000$, the optimum $\tilde{L}$ is 50 when $r' = 1$. If $r' = 10$, then $\tilde{L}$ becomes 16 ($\simeq \sqrt{5000/20}$) due to increase of comparisons at first stage.

By substituting $\tilde{L}$ into Equation (3.31), it becomes

$$\mathcal{T}(N) = \frac{N\sqrt{Nr'}}{\sqrt{2}} + \frac{N\sqrt{Nr'}}{\sqrt{2}} - \frac{N}{2} = \frac{N(2\sqrt{2Nr'}-1)}{2}, \tag{3.32}$$

which means that $O(N\sqrt{N})$ comparisons are needed for the constrained agglomerative clustering. Since the number of comparisons in the bisecting k-means clustering amounts to $O(N \log_2 N)$ (see

[27] If $y = f(x) + g(x)$, then $dy/dx = df(x)/dx + dg(x)/dx$.

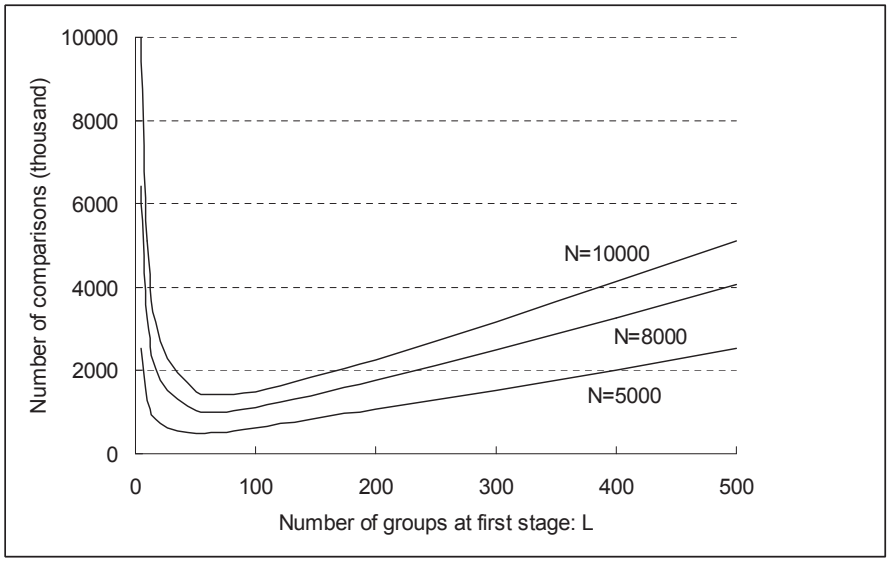


Figure 3.23: Complexity of constrained agglomerative clustering ($r' = 1$)

Equation (3.27)), computational complexity of the constrained agglomerative clustering increases faster than the bisecting k-means as N becomes larger, but slower than standard agglomerative hierarchical clustering with $O(N^2)$ comparisons between documents. For example, the ratio $(N\sqrt{N})/(N \log_2 N)$ is around 3.17 when $N = 1000$, but it becomes around 7.52 when $N = 10000$ and 19.04 when $N = 100000$ (note again that cluster generation is also needed for the agglomerative hierarchical clustering).

3.2.5 Other hierarchical document clustering

In *suffix tree clustering* (STC) [329, 330], a suffix tree constructed for the target document set is used for clustering. Namely, since each node of the tree is related with documents containing a string designated by the node, it is possible to consider a set of documents clinging to each node as a cluster, which is called 'base cluster'. A dendrogram can be constructed by merging iteratively the base clusters based on a similarity measure.

Agglomerative hierarchical clustering algorithm based on 'information bottleneck method' [286] tries to merge documents based on mutual information (MI) between document clusters and terms such that

$$M_I(\mathcal{C}, \Omega) = \sum_{k:C_k \in \mathcal{C}} \sum_{j:t_j \in \Omega} P(C_k)P(t_j|C_k) \times \log \frac{P(t_j|C_k)}{P(t_j)}, \tag{3.33}$$

where $\mathcal{C}$ is a set of clusters and Ω denotes a set of terms, as before. More specifically, when decrease of $M_I(\mathcal{C}, \Omega)$ caused by merging two clusters is smaller than by merging any other pairs, they are actually merged into a single cluster at the current stage (i.e., information lost by merging is the smallest), which allows to construct a dendrogram [28]. In addition, by exchanging roles of terms and documents in $M_I(\mathcal{C}, \Omega)$, term clusters $\tilde{T} = \{\tilde{T}_1, \ldots, \tilde{T}_{L'}\}$ (where $\tilde{T}_k$ indicates a set of terms and $k = 1, \ldots, L'$) can be obtained by using $d_i \in D$ as classification features. After that, documents can be clustered by computing $M_I(\mathcal{C}, \tilde{T})$ [286], which may be interpreted as a clustering technique based on more compact representations of documents.

[28] For instance, probabilities in the definition of $M_I(\mathcal{C}, \Omega)$ can be estimated such that $P(t_j|C_k) = P(C_k)^{-1} \sum_{i:d_i \in C_k} P(d_i)P(t_j|d_i)$, $P(t_j|d_i) = \mathrm{f}_{ij}/l_i$, $P(C_k) = \sum_{i:d_i \in C_k} P(d_i)$, $P(d_i) = N^{-1}$ and $P(t_j) = M^{-1}$ (see [286]).

3.3 Flat Partitioning

This section reviews techniques of flat partitioning, by which a cluster set $\{C_1, \ldots, C_L\}$ is directly driven. In general, the number of possible patterns in such partitioning amounts to

$$\frac{1}{L!}\sum_{k=1}^{L}(-1)^{L-k}\left(\begin{array}{c} L \\ k \end{array}\right)k^N, \tag{3.34}$$

which increases rapidly as N becomes larger (Webb(2002) [313], p.377). Therefore, it is important to develop clustering methods providing approximately an 'optimum' partition in real time. It should be noted that probabilistic and matrix-based clustering algorithms described in next chapter generate also a flat partition. In this section, traditional techniques for it are mainly focused on.

3.3.1 K-means algorithms for document clustering

The most typical method of flat partitioning is k-means algorithm, which allows to group documents in less computational complexity as already mentioned. Due to this advantage, the k-means algorithms have been often applied to DC problems.

Standard k-means algorithm

Figure 3.8 shows the basic k-means algorithm in batch mode, which is the simplest version of k-means methods. Unfortunately, an inappropriate set of initial seeds may generate 'bad' clusters as explained in Section 3.1.2. Table 3.10 indicates an experimental result of the basic k-means algorithm for the sample DB in Table 3.1 when $L = 3$. Because $N = 10$ and $L = 3$, there are different 120 patterns ($= {}_{10}C_3$) of seed sets in total from $\{d_1, d_2, d_3\}$ to $\{d_8, d_9, d_{10}\}$, and Table 3.10 summarizes clustering results from 120 runs in which each seed set was used, respectively. For example, 29 seed sets (24.2%) generated a good result, $\mathcal{C} = \{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$, with the smallest RSS J in Equation (3.8) (i.e., RSS is about 2.485). However, in other words, only 24.2% of total 120 patterns can bring the result with the minimum RSS, and other seed sets failed to generate it. This is a serious problem in using the basic k-means algorithm because often the initial seed set is randomly selected under the assumption that there is no knowledge on a 'good' seed set.

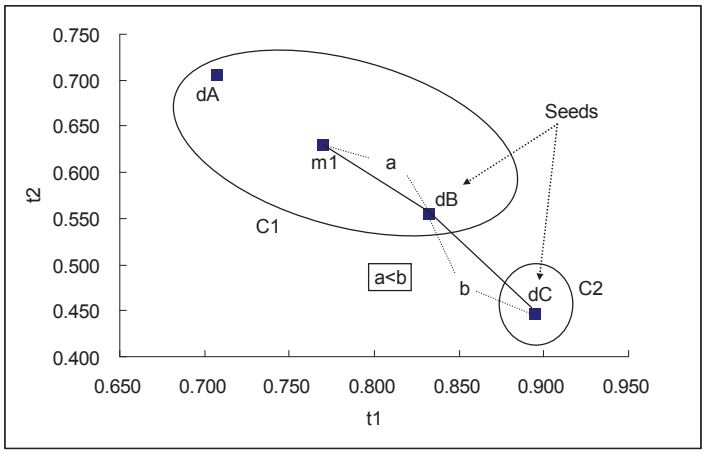


Figure 3.24: Example of k-means algorithm

One of the solutions would be to consider the RSS as a criterion of 'good' clustering, and to try to find computationally a partition that provides the minimum RSS. Namely, if the RSS of a particular partition $\mathcal{C} = \{C_1, \ldots, C_L\}$ is denoted by $J(\mathcal{C})$, then the purpose of flat partitioning of a document set is formulated as detecting $\mathcal{C}'$ such as $\mathcal{C}' = \arg\min_{\mathcal{C}} J(\mathcal{C})$. In general, $J(\mathcal{C})$ corresponds to an 'objective function' in the minimization problem. As mentioned above, there are a tremendously large number of ways of partitioning a set of N elements if N is not so small,

Table 3.10: Result of basic k-means algorithm: $L = 3$

RSS J	#	%	Resultant set of clusters
2.485	29	24.2%	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
2.583	1	0.8%	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7, d_{10}\}, \{d_6, d_8, d_9\}$
2.670	9	7.5%	$\{d_1, d_2, d_3, d_5\}, \{d_4, d_7, d_{10}\}, \{d_6, d_8, d_9\}$
2.729	14	11.7%	$\{d_1, d_2, d_3, d_5\}, \{d_4, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
2.930	6	5.0%	$\{d_1, d_2, d_3\}, \{d_4, d_7, d_{10}\}, \{d_5, d_6, d_8, d_9\}$
3.026	9	7.5%	$\{d_1, d_2, d_3, d_7\}, \{d_4, d_{10}\}, \{d_5, d_6, d_8, d_9\}$
3.200	1	0.8%	$\{d_1, d_2, d_3\}, \{d_4, d_{10}\}, \{d_5, d_6, d_7, d_8, d_9\}$
3.256	12	10.0%	$\{d_1, d_2, d_3, d_4, d_5, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}$
3.257	3	2.5%	$\{d_1, d_2, d_3\}, \{d_4, d_8, d_9, d_{10}\}, \{d_5, d_6, d_7\}$
3.272	4	3.3%	$\{d_1\}, \{d_2, d_3, d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
3.308	7	5.8%	$\{d_1, d_2, d_3\}, \{d_4, d_8, d_{10}\}, \{d_5, d_6, d_7, d_9\}$
3.383	7	5.8%	$\{d_1\}, \{d_2, d_3, d_5\}, \{d_4, d_6, d_7, d_8, d_9, d_{10}\}$
3.424	5	4.2%	$\{d_1\}, \{d_2, d_3, d_5, d_7\}, \{d_4, d_6, d_8, d_9, d_{10}\}$
3.471	6	5.0%	$\{d_1\}, \{d_2, d_3\}, \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$
3.510	3	2.5%	$\{d_1, d_2, d_3, d_5\}, \{d_4, d_8, d_{10}\}, \{d_6, d_7, d_9\}$
3.924	1	0.8%	$\{d_1, d_2, d_3, d_6, d_8, d_9\}, \{d_4, d_5, d_7\}, \{d_{10}\}$
3.970	1	0.8%	$\{d_1\}, \{d_2, d_3, d_5, d_6, d_8, d_9\}, \{d_4, d_7, d_{10}\}$
4.060	1	0.8%	$\{d_1, d_2, d_3, d_5, d_6, d_8, d_9\}, \{d_4, d_7\}, \{d_{10}\}$
4.158	1	0.8%	$\{d_1, d_2, d_3, d_6, d_8, d_9\}, \{d_4, d_{10}\}, \{d_5, d_7\}$
Total	120	100%	

Note: # means the number of seed sets.

and it is usually impossible to compute values of $J(\mathcal{C})$ for all possible patterns of the partition. To solve the problem, an 'iterative optimization' is usually used for the partition.

Suppose that there are three documents whose vectors are $\mathbf{d}_A = [1, 1]^T$, $\mathbf{d}_B = [3, 2]^T$ and $\mathbf{d}_C = [6, 3]^T$ ($M = 2$). If unit vectors are computed as $\tilde{\mathbf{d}}_i = \mathbf{d}_i / \|\mathbf{d}_i\|$, then it becomes that $\tilde{\mathbf{d}}_A = [0.707, 0.707]^T$, $\tilde{\mathbf{d}}_B = [0.832, 0.555]^T$ and $\tilde{\mathbf{d}}_C = [0.894, 0.447]^T$. The RSS of each partition can be computed as $J(\{\{d_A, d_B\}, d_C\}) = 0.019$, $J(\{d_A, \{d_B, d_C\}\}) = 0.008$, and $J(\{\{d_A, d_C\}, d_B\}) = 0.051$, which means that $\{d_A, \{d_B, d_C\}\}$ is the best partition based on the criterion J in the case that $L = 2$. If the basic k-means algorithm is applied to them with two seeds d_B and d_C and it works in order of $d_A \to d_B \to d_C$, then at first iteration, $C_1 = \{d_A, d_B\}$ and $C_2 = \{d_C\}$ are generated since d_A is near to seed d_B (see Figure 3.24). Unfortunately, since $\|\tilde{\mathbf{d}}_B - \mathbf{m}_1\| = 0.099$ and $\|\tilde{\mathbf{d}}_B - \mathbf{m}_2\| = 0.124$, document d_B does not move to C_2 in next iteration, and therefore, the best partition can not be obtained by the basic k-means algorithm. However, in such case, by 'explicitly' computing J after tentatively transferring d_B from C_1 to C_2, the best partition $\{d_A, \{d_B, d_C\}\}$ can be automatically detected. If d_i moves tentatively from $C_{k'}$ to C_k, then $\mathbf{m}_k$ changes to

$$\mathbf{m}_k^* = \frac{\tilde{n}_k \mathbf{m}_k + \tilde{\mathbf{d}}_i}{\tilde{n}_k + 1} = \left(1 - \frac{1}{\tilde{n}_k + 1}\right) \mathbf{m}_k + \frac{\tilde{\mathbf{d}}_i}{\tilde{n}_k + 1} = \mathbf{m}_k + \frac{\tilde{\mathbf{d}}_i - \mathbf{m}_k}{\tilde{n}_k + 1} \tag{3.35}$$

and J_k in Equation (3.5) increases to

$$\begin{aligned}
J_k^* &= \sum_{i': d_{i'} \in C_k} \|\tilde{\mathbf{d}}_{i'} - \mathbf{m}_k^*\|^2 + \|\tilde{\mathbf{d}}_i - \mathbf{m}_k^*\|^2 \\
&= \sum_{i': d_{i'} \in C_k} \left\| \tilde{\mathbf{d}}_{i'} - \mathbf{m}_k - \frac{\tilde{\mathbf{d}}_i - \mathbf{m}_k}{\tilde{n}_k + 1} \right\|^2 + \left\| \frac{\tilde{n}_k}{\tilde{n}_k + 1} (\tilde{\mathbf{d}}_i - \mathbf{m}_k) \right\|^2 \\
&= \sum_{i': d_{i'} \in C_k} \|\tilde{\mathbf{d}}_{i'} - \mathbf{m}_k\|^2 + \tilde{n}_k \left\| \frac{\tilde{\mathbf{d}}_i - \mathbf{m}_k}{\tilde{n}_k + 1} \right\|^2 + \left\| \frac{\tilde{n}_k}{\tilde{n}_k + 1} (\tilde{\mathbf{d}}_i - \mathbf{m}_k) \right\|^2 \\
&= J_k + \frac{\tilde{n}_k + \tilde{n}_k^2}{(\tilde{n}_k + 1)^2} \|\tilde{\mathbf{d}}_i - \mathbf{m}_k\|^2 = J_k + \frac{\tilde{n}_k}{\tilde{n}_k + 1} \|\tilde{\mathbf{d}}_i - \mathbf{m}_k\|^2.
\end{aligned} \tag{3.36}$$

Table 3.11: Result of Hartigan-Wong algorithm ($L = 3$)

RSS J	#	%	Set of clusters
2.485	106	88.3%	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
3.383	14	11.7%	$\{d_1\}, \{d_2, d_3, d_5\}, \{d_4, d_6, d_7, d_8, d_9, d_{10}\}$
Total	120	100%	

Note: # denotes the number of seed sets.

Similarly, if $n_{k'} > 2$, then $J_{k'}$ decreases to

$$J^*_{k'} = J_{k'} - \frac{\tilde{n}_{k'}}{\tilde{n}_{k'} - 1} \|\tilde{\mathbf{d}}_i - \mathbf{m}_{k'}\|^2, \tag{3.37}$$

by the transfer. Therefore, the transfer of d_i from $C_{k'}$ to C_k is effective for finding $\mathcal{C}' = \arg\min_{\mathcal{C}} J(\mathcal{C})$ if $|J^*_{k'} - J_{k'}| > |J^*_k - J_k|$.

For convenience, the amount of decrease or increase by the transfer is written such that

$$\rho_{k'}(k) = \begin{cases} \frac{\tilde{n}_k}{\tilde{n}_k + 1} \|\tilde{\mathbf{d}}_i - \mathbf{m}_k\|^2 & k \neq k' \\ \frac{\tilde{n}_{k'}}{\tilde{n}_{k'} - 1} \|\tilde{\mathbf{d}}_i - \mathbf{m}_{k'}\|^2 & k = k' \end{cases} . \tag{3.38}$$

Thus document d_i in $C_{k'}$ can be reassigned to C_K where $K = \arg\min_k \rho_{k'}(k)$ in the reallocation (or transfer) stage after the basic k-means clustering. If $K = k'$, then any reallocation does not occur. When a document is reallocated to another cluster, $\mathbf{m}_{k'}$ and $\mathbf{m}_K$ have to be updated. The reallocation is iteratively continued until the resulting clusters become stable [29].

An efficient algorithm based on the reallocation (or transfer) that decreases iteratively the RSS has been proposed by Hartigan & Wong(1979) [119], which is widely well-known as an effective k-means algorithm [30]. The iterative reallocation procedure works better than the basic k-means algorithm, which is clear from the above example. However, it should be noted that such kind of iterative optimization may find 'local' minimum of the objective function (i.e., RSS), not 'global' one (see Figure 3.25). Actually, like the basic k-means algorithm, there would be initial seed sets reaching to the local minimum. For example, Table 3.11 shows clustering results of the Hartigan-Wong algorithm under the same experimental setting with Table 3.10, which indicates about 12% of seed sets falls into a local minimum (RSS=3.383) although over 88% reaches successfully to the global minimum (RSS=2.485) for the sample DB.

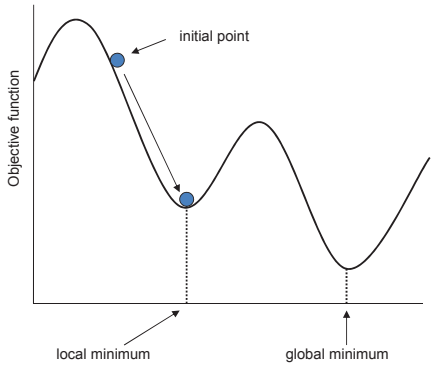


Figure 3.25: Local and global mimimums

The search of documents to be transfered in the reallocation stage may increase slightly its computational steps. However, in the case of Hartigan-Wong algorithm, the computational complexity remains to be $O(NLMr)$ (see also Section 3.1.2). Actually, in the experiments using the

[29] The reallocation-based clustering may sometimes be called 'iterative minimum-squared-error' clustering [84].
[30] Actually, the default algorithm of the kmeans( ) function in R [246] is the Hartigan-Wong algorithm.

sample DB, results of which are shown in Table 3.11, an average of the total number of checked documents for each run amounts to 53.37, which means that $r = 5.337$ on average over 120 runs for the sample DB. Since $r = 2.475$ on average in the basic k-means algorithm, computational complexity of the Hartigan-Wong algorithm was about 2.16 times of that of the basic k-means algorithm for the sample DB.

Online k-means algorithm

For executing efficiently iterative computation of the k-means algorithm, it is desirable to store all document vectors in main memory. Unfortunately, if memory size is not sufficiently large and the vectors can not be placed on it, then the file on HDD has to be repeatedly accessed for reading out each document in the iterative computation, which would make the processing very insufficient. A simple way for avoiding inefficiently repeated access to HDD is to execute the k-means algorithm without any iteration, which can be considered as a single-pass clustering operation. Although quality of the clustering result may be deteriorated, processing time would be drastically improved.

Similarly, the 'single-pass' operation is also relevant to *online document clustering* in which each document arriving successively has to be sequentially assigned to a cluster under an assumption that detailed information on vectors of past documents previously processed is unavailable for some reasons [31]. In this situation, only cluster representatives (e.g., centroids) before the target document arrives can be used, and therefore, iterative computation is usually impossible. Figure 3.26 shows a basic procedure of the *online k-means clustering*. If centroid vector $\mathbf{m}_k$ is adopted as cluster representative $\mathbf{c}_k$, then the initial cluster vector is set as $\mathbf{m}_k = \tilde{\mathbf{d}}_k$ $(k = 1, \ldots, L)$ at the step 1) in Figure 3.26 and the cluster vector can be updated using d_i $(i = L+1, \ldots, N)$ such that

$$\mathbf{m}_k \leftarrow \frac{1}{\tilde{n}_k + 1}(\tilde{n}_k \mathbf{m}_k + \tilde{\mathbf{d}}_i) \tag{3.39}$$

(and after that, $\tilde{n}_k \leftarrow \tilde{n}_k + 1$ at the step 2)).

‡ Online k-means algorithm for DC

Set: The number of clusters L

1) For $i = 1, \ldots, L$, allocate automatically document d_i to cluster C_i and set cluster vector $\mathbf{c}_i$ based on $\tilde{\mathbf{d}}_i$.

2) For $i = L+1, \ldots, N$, allocate document d_i to the nearest cluster C_k and concurrently update cluster vectors $\mathbf{c}_k$ based on $\tilde{\mathbf{d}}_i$.

Out: Clusters $C_1, \ldots, C_L$ and final vectors $\mathbf{c}_1, ..., \mathbf{c}_L$.

Figure 3.26: Online k-means algorithm for document clustering

Whereas in the basic k-means algorithm that is a 'batch mode' clustering, updating of cluster vectors is done all together after N documents are allocated, in online k-means algorithm, cluster vectors have to be updated immediately after each document is allocated to a cluster, which is often called 'incremental'. Thus in the online or *incremental clustering*, it is possible that each assignment of a document influences directly that of next documents, which means that the processing order of documents has a critical effect on clustering result.

Also, quality of online k-means clustering is largely dependent on the initial L documents used as seeds for clustering. Suppose that documents in the sample DB will arrive in turn from d_1 to d_{10}. If $L = 3$, then d_1, d_2 and d_3 are selected as seeds, which clearly leads to a poor result because they intrinsically constitute a single cluster separated from other seven documents (see Table 3.11).

[31] For example, in order to find a 'novel' topic among news stories arriving successively, a stream of news stories has to be effectively clustered. This is a kind of 'text stream clustering', which is discussed in Section 3.3.7.

Actually, the online k-means clustering for the data provides an invalid grouping,

$$\mathcal{C} = \{\{d_1, d_4, d_7, d_{10}\}, \{d_2\}, \{d_3, d_5, d_6, d_8, d_9\}\}$$

with RSS=4.451. If the order of processing documents was changed to a sequence of d_8, d_5, d_1, d_9, d_4, d_6, d_2, d_7, d_3 and d_{10} (i.e., d_8, d_5 and d_1 are seeds), then the best clustering result,

$$\mathcal{C} = \{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$$

with RSS=2.485, was obtained by the online k-means method. Figure 3.27 shows an experimental result of executing repeatedly 100000 times the online k-means clustering with changing the order of documents to be processed, in which the order for each run was determined randomly by using Math.random() method of Java language. As shown in Figure 3.27, about 18.4% of 100000 runs produced the above best result, which is lower than 24.2% by the basic k-means algorithm (see Table 3.10). Also, the RSS of the worst case in the online k-means clustering was 5.061, which is larger than 4.158 in the basic k-means algorithm. Although the online k-means clustering is an efficient single-pass method with $r = 1$, its effectiveness becomes inevitably lower than that of batch mode algorithms with multiple iterations.

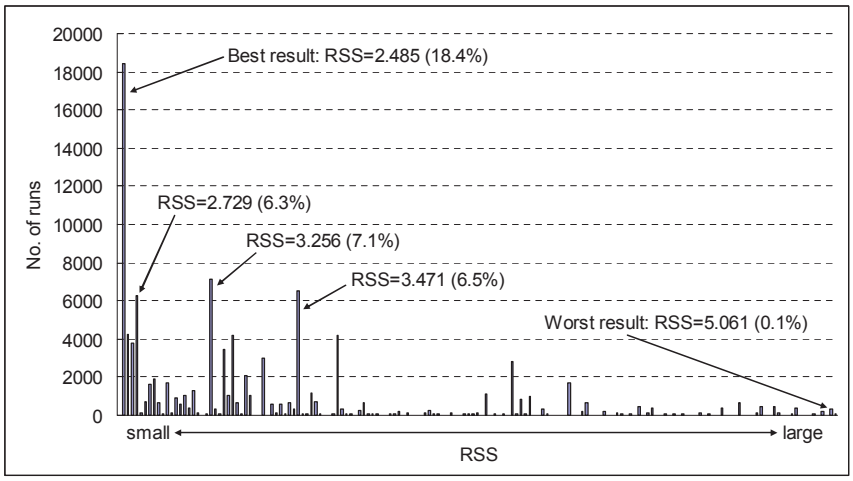


Figure 3.27: Experimental results of online k-means clustering (100000 trials)

Scalable k-means algorithm

When all document vectors can not be stored in main memory, a method for enabling multiple iterations is to select randomly some documents as a sample and to apply a batch mode k-means algorithm to only the sample stored in the memory. If 'valid' cluster representatives are constructed from the sample, then in next step, it is enough to read out each document from the file on HDD, and to assign them to the nearest cluster individually (i.e., 'single-pass' in this part), in which all document vectors are not necessarily on the memory.

Furthermore, in order to remove influence from sampling error, it is possible to repeat the clustering for some different samples, and to merge information from each sample [37, 90]. A simple *scalable k-means* algorithm [340] according to this idea is shown in Figure 3.28 where 'history vectors' keeping information on cluster centroids obtained from previous samples are denoted by $\mathbf{h}_1, \ldots, \mathbf{h}_L$, and $\dot{n}_k$ indicates the cumulative number of documents assigned to cluster C_k in the previous samples ($k = 1, \ldots, L$) [32]. In this algorithm, previous results of an iterative k-means algorithm are consecutively passed to next clustering through the history vectors. More precisely, the history vector is incorporated into next sample as a single document vector (see step 2)). When a history vector is assigned to a cluster at step 2) of the algorithm, its centroid vector is updated

[32] Note that the original algorithm [340] is slightly changed here.

with consideration of documents in the previous samples. For instance, if using the basic k-means algorithm, then it is feasible to update the centroid vector as a weighted average such that

$$\mathbf{m}_k \leftarrow \frac{1}{\tilde{n}_k + \dot{n}_{k'}}(\tilde{n}_k \mathbf{m}_k + \dot{n}_{k'} \mathbf{h}_{k'}), \tag{3.40}$$

when history vector $\mathbf{h}_{k'}$ is assigned to cluster C_k at step 2) in the procedure (simultaneously, the number of documents in the cluster is increased as $\tilde{n}_k \leftarrow \tilde{n}_k + \dot{n}_{k'}$).

‡ Simple scalable k-means algorithm for DC

Set: The number of clusters L and sample size n ($n < N$)

1) Read vectors of $d_1, \ldots, d_n$ into the memory, and execute an iterative k-means algorithm for n documents (the number of documents in each cluster is recorded as $\dot{n}_k$ and each final centroid vector is kept into $\mathbf{h}_k$ where $k = 1, \ldots, L$). Set s to n.

2) Read vectors of $d_{1+s}, \ldots, d_{n+s}$ into memory and execute the iterative k-means algorithm for $\{\tilde{\mathbf{d}}_{1+s}, \ldots, \tilde{\mathbf{d}}_{n+s}, \mathbf{h}_1, \ldots, \mathbf{h}_L\}$ using weighted average based on $\dot{n}_k$ (if $n + s > N$, then $d_{N+1}, d_{N+2}, \ldots$ are inevitably ignored).

3) Update $\mathbf{h}_k$ as $\mathbf{h}_k \leftarrow \mathbf{m}_k / \|\mathbf{m}_k\|$, and change $\dot{n}_k$ according to the clustering result ($k = 1, \ldots, L$).

4) If $n + s \geq N$, then terminate the procedure. Otherwise, $s \leftarrow n + s$ and return to 2).

Out: Clusters $C_1, \ldots, C_L$ and final centroid vectors $\mathbf{m}_1, ..., \mathbf{m}_L$.

Figure 3.28: Simple scalable k-means algorithm for document clustering

Another technique of scalable k-means clustering is to constitute 'sub-clusters' in the first scan of the target file, and to group the sub-clusters into L 'global clusters' iteratively by a batch mode k-means algorithm at the second stage [164] [33]. If all vectors of the sub-clusters can be stored in main memory, then it is enough to scan the file on HDD just once. In order to generate the sub-clusters at the first stage, the leader-follower clustering algorithm (Section 3.3.2) may be available. A sub-cluster and its centroid are denoted here by $\ddot{C}_h$ and $\ddot{\mathbf{m}}_h$, respectively. Note that the centroid of a cluster at the second stage can be computed as

$$\mathbf{m}_k = \frac{1}{\tilde{n}_k} \sum_{h:\ddot{C}_h \subset C_k} \ddot{n}_h \ddot{\mathbf{m}}_h, \tag{3.41}$$

where $\ddot{n}_h$ indicates the number of documents in $\ddot{C}_h$, and $\tilde{n}_k = \sum_{h:\ddot{C}_h \subset C_k} \ddot{n}_h$ ($k = 1, \ldots, L$). Similarly, other statistics such as RSS have to be calculated with considering the number of documents belonging to the sub-cluster.

Spherical k-means algorithm

As described in previous chapters, inner product of two document vectors is often used for measuring their similarity according to the traditional IR theory (i.e., vector space model). If document vectors are normalized such that $||\mathbf{d}_i|| = 1$, then clustering results obtained by the Euclidean distance would be almost same with those by the inner product [34]. However, in terms of efficiency, the inner product has a computational advantage when document vectors are high dimensional and sparse because it is enough to look into common terms appearing in both the two documents

[33] In [164], this technique is called 'BIRCH-like k-means'. The BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [334] is a general algorithm for clustering large-scale datasets.

[34] As described in Section 3.1.1, the squared Euclidean distance between two unit vectors is 'mathematically' coincident with their similarity computed as the cosine coefficient.

for computing it. Meanwhile, for computing the Euclidean distance, terms included only in the one document have to be additionally considered. For example, in the case of d_1 and d_5 of the sample DB, since

$$\tilde{\mathbf{d}}_1 = [0.970, 0.243, 0.000, 0.000, 0.000, 0.000]^T$$
$$\tilde{\mathbf{d}}_5 = [0.000, 0.667, 0.667, 0.000, 0.333, 0.000]^T,$$

the inner product can be computed from only the weight of t_2 whereas those of t_1, t_3 and t_5 are also needed for calculation of the Euclidean distance. The computational efficiency of the inner product leads to using it instead of the Euclidean distance in determining a cluster to which each document belongs in k-means algorithms, which is often called *spherical k-means* method [81] [35].

Actually, in the the spherical k-means method, document d_i is allocated to C_k such that

$$k = \arg\max_{k'=1,\ldots,L} \tilde{\mathbf{d}}_i^T \frac{\mathbf{c}_{k'}}{\|\mathbf{c}_{k'}\|}, \tag{3.42}$$

where $\mathbf{c}_k$ denotes a cluster vector, which is typically assumed that

$$\mathbf{c}_k = \mathbf{a}_k \equiv \sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i. \tag{3.43}$$

Since $\tilde{\mathbf{d}}_i$ and $\mathbf{c}_k/\|\mathbf{c}_k\|$ are unit vectors, inner product in Equation (3.42) is equivalent to the cosine coefficient between $\mathbf{c}_k$ and $\mathbf{d}_i$.

Straightforwardly, it is feasible to implement a spherical k-means algorithm by using Equation (3.42) for allocation of documents to clusters and by updating cluster vectors based on Equation (3.43) in procedure of the basic k-means method. In the case of a spherical k-means method optimizing an objective function, J_k defined in Equation (3.5) is replaced by

$$\tilde{J}_k = \sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i^T \mathbf{a}_k / \|\mathbf{a}_k\|, \tag{3.44}$$

and $\sum_k \tilde{J}_k$ is maximized in iterative computation [295] [36]. When new document vector $\tilde{\mathbf{d}}_\dagger$ is incorporated into a vector of cluster C_k, increment of $\tilde{J}_k$ (denoted by $\triangle_+ \tilde{J}_k$) is written as

$$\triangle_+ \tilde{J}_k = \sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i^T \frac{\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} + \tilde{\mathbf{d}}_\dagger^T \frac{\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} - \tilde{J}_k. \tag{3.45}$$

Since

$$\begin{aligned} \sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i^T \frac{\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} &= \frac{1}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} \left(\sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i^T \mathbf{a}_k + \sum_{i:d_i \in C_k} \tilde{\mathbf{d}}_i^T \tilde{\mathbf{d}}_\dagger \right) \\ &= \frac{\|\mathbf{a}_k\|}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} \left(\tilde{J}_k + \frac{{\mathbf{a}_k}^T \tilde{\mathbf{d}}_\dagger}{\|\mathbf{a}_k\|} \right), \end{aligned} \tag{3.46}$$

it becomes that

$$\triangle_+ \tilde{J}_k = \left(\frac{\|\mathbf{a}_k\|}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|} - 1 \right) \tilde{J}_k + \frac{2\tilde{\mathbf{d}}_\dagger^T \mathbf{a}_k + 1}{\|\mathbf{a}_k + \tilde{\mathbf{d}}_\dagger\|}. \tag{3.47}$$

Similarly, if document $d_\dagger$ ($\in C_k$) is removed from C_k, then decrement of $\tilde{J}_k$ is represented as

$$\triangle_- \tilde{J}_k = \left(1 - \frac{\|\mathbf{a}_k\|}{\|\mathbf{a}_k - \tilde{\mathbf{d}}_\dagger\|} \right) \tilde{J}_k + \frac{2\tilde{\mathbf{d}}_\dagger^T \mathbf{a}_k - 1}{\|\mathbf{a}_k - \tilde{\mathbf{d}}_\dagger\|}. \tag{3.48}$$

Therefore, by incorporating directly Equations (3.47) and (3.48) into the Hartigan-Wong algorithm [119], the spherical k-means method based on the objective function can be executed [37].

[35] The spherical k-means algorithm can be formalized as a method based on a probabilistic mixture model using von Mises-Fisher distribution (see Section 4.1.1).

[36] An experiment for comparing various criteria including $\tilde{J}_k$ was reported by [338].

[37] Of course, computations of the Euclidean distance are replaced by those of inner product in the original algorithm. Note that if $d_\dagger \in C_{k'}$ and $\triangle_- \tilde{J}_{k'} > \triangle_+ \tilde{J}_k$ for any cluster C_k ($k \neq k'$), then $d_\dagger$ does not move to the other cluster. This technique was used in an experiment reported in Section 5.3.

Willett's algorithm

In order to execute k-means algorithms, the number of clusters or a set of initial cluster vectors (i.e., seeds) has to be given a priori by users. However, they are often unknown in DC situations. Willett's algorithm [317] works without any information on the number of clusters or seeds by using an index file.

In the algorithm, an initial cluster vector is computed from a set of documents sharing an index term, which is easily known by checking an index file (Section 3.2.2). For example, in the case of the sample DB in Table 3.1, a cluster vector for t_1 is constructed from $\{d_1, d_2, d_3\}$ that is registered in the index file as an entry of t_1. Since such cluster is created for every index term, the number of clusters is M (i.e., $L = M$) in initial stage ($L = 6$ in the sample DB), and it is possible that a single document contributes to computation of two or more cluster vectors.

Next, every document is allocated to a single cluster and its cluster vector is updated in a manner of k-means algorithms. After all documents were processed, clusters to which no document is allocated are deleted. The process of allocating, updating and deleting is repeated until a condition for termination is satisfied.

Users of this algorithm do not have to input L or seeds, which is an advantage. However, if $N \simeq M$, then the number of comparisons between documents and clusters approaches to N^2. Also, it is difficult to predict the total number of iterations r providing enough results. Thus feature selection and definition of terminating condition would play a key role for executing the Willett's algorithm.

SKWIC algorithm

Another objective function,

$$J_S = \sum_{k=1}^{L} \sum_{i:d_i \in C_k} \sum_{j=1}^{M} \tilde{w}_{kj} \hat{w}_{ijk} + \sum_{k=1}^{L} \left(\delta_k \sum_{j=1}^{M} \tilde{w}_{kj}^2 \right), \tag{3.49}$$

was used in *SKWIC* (simultaneous keyword identification and clustering of text document) algorithm [96] where

$$\hat{w}_{ijk} = \frac{1}{M} - w_{ij} m_{kj}, \tag{3.50}$$

and m_{kj} is j-th element of centroid vector $\mathbf{m}_k$ in Equation (3.6) (δ_k is a parameter for C_k). The SKWIC is a variation of SCAD (simultaneous clustering and attribute discrimination) [95], in which clustering and feature weighting are performed simultaneously so as to minimize the objective function [38]. More specifically, it is attempted to obtain concurrently optimal clusters and optimal term weights in clusters, denoted by $\tilde{w}_{kj}$ ($k = 1, \ldots, L; j = 1, \ldots, M$), under a condition that

$$\sum_{j=1}^{M} \tilde{w}_{kj} = 1, \quad k = 1, ..., L \tag{3.51}$$

(i.e., m_{kj} and $\tilde{w}_{kj}$ are completely different in the algortihm).

Note that $\sum_j \tilde{w}_{kj} \hat{w}_{ijk}$ in Equation (3.49) measures 'dissimilarity' between a document and a cluster since term weight $\hat{w}_{ijk}$ in a document becomes small as $w_{ij} \times m_{kj}$ is larger. The dissimilarity is computed as an inner product, and therefore, it would be better that each document vector is normalized such that $\tilde{\mathbf{d}}_i = \mathbf{d}_i / \|\mathbf{d}_i\|$ as in Equation (3.2).

The objective function in Equation (3.49) includes two components. The first component would possibly become smaller when only fewer terms are relevant to each cluster. Thus it can be interpreted that the first component measures 'compactness' of clusters. In contrast, the second component is the sum of squared term weights, which becomes minimum when all weights are equal. Balance of the two components is adjusted by parameters δ_k ($k = 1, \ldots, L$), which is

[38] In 'co-clustering', documents (i.e., objects) and terms (i.e., features) are concurrently grouped (see also Section 4.3.2).

defined as a ratio of the first and second components by each cluster,

$$\delta_k = c_\delta \frac{\sum_{i:d_i \in C_k} \sum_{j=1}^{M} \tilde{w}_{kj} \hat{w}_{ijk}}{\sum_{j=1}^{M} \tilde{w}_{kj}^2}, \quad k = 1, \ldots, L, \tag{3.52}$$

where c_δ is a constant [96].

By using the Lagrange multiplier technique [39], it is feasible to obtain term weights $\tilde{w}_{kj}$ minimizing the objective function in Equation (3.49). Namely, if putting that $\mathcal{H} = J_S - \sum_k \lambda_k (\sum_j \tilde{w}_{kj} - 1)$ based on the condition in Equation (3.51) where λ_k is the Lagrange multiplier for C_k, then

$$\frac{\partial \mathcal{H}}{\partial \tilde{w}_{kj}} = \sum_{i:d_i \in C_k} \hat{w}_{ijk} + 2\delta_k \tilde{w}_{kj} - \lambda_k. \tag{3.53}$$

Therefore, by solving $\partial \mathcal{H} / \partial \tilde{w}_{kj} = 0$, it follows that

$$\tilde{w}_{kj} = \frac{1}{2\delta_k} \left(\lambda_k - \sum_{i:d_i \in C_k} \hat{w}_{ijk} \right). \tag{3.54}$$

Equation (3.51) requires that $\sum_j (2\delta_k)^{-1} (\lambda_k - \sum_i \hat{w}_{ijk}) = 1$, which leads to $\lambda_k = M^{-1}(2\delta_k + \sum_j \sum_i \hat{w}_{ijk})$. By substituting it into Equation (3.54),

$$\tilde{w}_{kj} = \frac{1}{M} + \frac{1}{2\delta_k} \sum_{i:d_i \in C_k} \left[\left(\frac{1}{M} \sum_{j=1}^{M} \hat{w}_{ijk} \right) - \hat{w}_{ijk} \right], \tag{3.55}$$

is finally obtained.

Procedure of the SKWIC algorithm is explained in Figure 3.29, and Tables 3.12 and 3.13 show a result of applying the algorithm to the sample DB. In the execution, a document vector was defined as a unit vector based on tf (i.e., $w_{ij} = \mathrm{f}_{ij} / \sqrt{\mathrm{f}_{i1}^2 + \ldots + \mathrm{f}_{iM}^2}$), and $\tilde{\mathbf{d}}_1$, $\tilde{\mathbf{d}}_2$ and $\tilde{\mathbf{d}}_3$ were selected as initial cluster centroids (note that the combination of d_1, d_2 and d_3 is never a good initial set) and $c_\delta = 10$.

As Table 3.12 shows, final clusters $C_1 = \{d_1, d_2, d_3\}$, $C_2 = \{d_6, d_8, d_9, d_{10}\}$ and $C_3 = \{d_4, d_5, d_7\}$, which are the same with those by the k-means algorithm (see Table 3.4), were obtained in 3rd iteration ($s = 3$). Also, parameter δ_k converges almost in 7th iteration, which means that values of $\tilde{w}_{kj}$ become also stable (see Equation (3.52)). The values are listed in Table 3.13, which shows that t_2 is the most relevant term to C_1, t_5 to C_2 and t_3 to C_3. The result can be intuitively interpreted as valid with referring to tf values in Table 3.1.

It would be possible to use scores of $\tilde{w}_{kj}$ for determining if a term is selected as a cluster label, which is an advantage of the SKWIC algorithm. However, selecting an appropriate value of c_δ may be difficult. For example, when $c_\delta = 1$, the final set of clusters becomes $C_1 = \{d_1, d_2, d_3, d_5\}$, $C_2 = \emptyset$ and $C_3 = \{d_4, d_6, d_7, d_8, d_9, d_{10}\}$, which is an invalid result for $L = 3$.

3.3.2 Leader-follower clustering algorithm

Whereas the number of clusters has to be given a priori for executing k-means algorithms, leader-follower clustering algorithm [84] works based on a predefined threshold θ_s of similarity between a document and a cluster, and the number of clusters L is determined posteriorly. A standard procedure of the leader-follower clustering algorithm is shown in Figure 3.30 [247]. In the algorithm, after computing similarities between each document and current clusters existing at the time, if its maximum degree exceeds θ_s, then the document is allocated to the most similar cluster [40]. Otherwise, the document is registered as a new cluster (i.e., a singleton). Therefore, the number

[39] In order to minimize (or maximize) an objective function $f(x)$ under a condition $g(x) = a$ where a denotes a constant, it is necessary to solve equations $\partial \mathcal{H} / \partial x = 0$ and $\partial \mathcal{H} / \partial \lambda = 0$ for a function $\mathcal{H} = f(x) - \lambda(g(x) - a)$. Usually, λ is called the Lagrange multiplier. Note that ∂ indicates partial differentiation. When there are two conditions, the function becomes $\mathcal{H} = f(x) - \lambda_1(g_1(x) - a_1) - \lambda_2(g_2(x) - a_2)$.

[40] This is exclusive clustering. When the document is allocated to all clusters whose similarity is over θ_s, nonexclusive clusters are obtained.

‡ SKWIC algorithm

Set: The number of clusters L and a constant c_δ.

1) Draw randomly L documents and record their vectors as centroids of L initial clusters. Also, initialize $\tilde{w}_{kj}$ such that $\tilde{w}_{kj} = 1/M$ $(k = 1, \ldots, L;\ j = 1, \ldots, M)$.

2) Compute $\hat{w}_{ijk}$ by Equation (3.50) $(i = 1, ..., N;\ j = 1, ..., M;\ k = 1, ..., L)$ and δ_k by Equation (3.52) $(k = 1, ..., L)$.

3) Update $\tilde{w}_{kj}$ by Equation (3.55) $(k = 1, \ldots, L;\ j = 1, \ldots, M)$.

4) Allocate d_i to k'-th cluster such as $k' = \arg\min_k \sum_j \tilde{w}_{kj}\hat{w}_{ijk}$ for $i = 1, \ldots, N$, and update each cluster centroid by Equation (3.6) based on the allocation (if $\tilde{w}_{kj} = 0$, then $m_{kj} = 0$).

5) If a condition is satisfied, then the procedure is terminated. Otherwise, return to 2).

Out: Clusters $C_1, \ldots, C_L$ and final weights $\tilde{w}_{kj}$ $(k = 1, \ldots, L$ and $j = 1, \ldots, M)$.

Figure 3.29: SKWIC algorithm

Table 3.12: Result of SKWIC for sample DB (1) ($L = 3$)

	Allocated cluster number in s-th iteration						
	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$
d_1	1	1	1	1	1	1	1
d_2	2	1	1	1	1	1	1
d_3	3	3	1	1	1	1	1
d_4	2	3	3	3	3	3	3
d_5	3	3	3	3	3	3	3
d_6	3	3	2	2	2	2	2
d_7	3	3	3	3	3	3	3
d_8	2	2	2	2	2	2	2
d_9	2	2	2	2	2	2	2
d_{10}	2	2	2	2	2	2	2
	Values of δ_k in s-th iteration						
	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$
C_1	77.38	77.37	70.16	67.58	67.50	67.50	67.50
C_2	61.17	65.12	63.33	60.40	60.28	60.27	60.27
C_3	61.87	56.41	58.65	57.69	57.66	57.66	57.66

Note: Initial cluster centroids are $\tilde{\mathbf{d}}_1$, $\tilde{\mathbf{d}}_2$ and $\tilde{\mathbf{d}}_3$, and $c_\delta = 10$.

Table 3.13: Result of SKWIC for sample DB (2) ($L = 3$)

	Term weight $\tilde{w}_{kj}$						
	t_1	t_2	t_3	t_4	t_5	t_6	Total
C_1	<u>0.1681</u>	<u>0.1803</u>	0.1629	0.1629	0.1629	0.1629	1.000
C_2	0.1615	0.1628	0.1626	<u>0.1669</u>	<u>0.1831</u>	<u>0.1631</u>	1.000
C_3	0.1608	0.1688	<u>0.1789</u>	0.1641	0.1661	0.1613	1.000

Note: These weights are of 7th iteration ($s = 7$).

Table 3.14: Result of leader-follower clustering for sample DB (1)

θ_s		Ascending order: $d_1 \rightarrow d_{10}$
0.2	$L = 2$	$\{d_1, d_2, d_3, d_5, d_6, d_8, d_9\}, \{d_4, d_7, d_{10}\}$
0.3	$L = 3$	$\{d_1, d_2, d_3, d_5\}, \{d_4, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
0.4	$L = 4$	$\{d_1, d_2, d_3, d_5\}, \{d_4, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}$
θ_s		Descending order: $d_{10} \rightarrow d_1$
0.2	$L = 2$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$
0.3	$L = 3$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
0.4	$L = 4$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}$

Note: Single-pass clustering.

of clusters L will be automatically determined after checking all documents. Actually, the value of L is highly dependent on θ_s and the degree of topical homogeneity in the document set. For instance, as θ_s is lower, L tends to be smaller because merging of documents into existing clusters becomes easy.

‡ Basic leader-follower clustering algorithm (for exclusive clustering)

Set: Similarity threshold θ_s.

1) Set first document d_1 as first cluster ($\mathbf{c}_1 \leftarrow \mathbf{d}_1$), and $L \leftarrow 1$ and $i \leftarrow 2$.

2) Compute similarities between $\mathbf{d}_i$ and vectors of all current clusters $\mathbf{c}_k$ ($k = 1, \ldots, L$). If the maximum of similarities $s(\mathbf{d}_i, \mathbf{c}_{k'})$ (i.e., $k' = \arg\max_k s(\mathbf{d}_i, \mathbf{c}_k)$) exceeds θ_s, then allocate document d_i to cluster $C_{k'}$ and update $\mathbf{c}_{k'}$ based on the allocation. Otherwise, generate a new cluster such that $L \leftarrow L + 1$ and $\mathbf{c}_L \leftarrow \mathbf{d}_i$.

3) $i \leftarrow i + 1$. If $i > N$, then terminate the procedure. Otherwise return to 2).

Out: The number of clusters L, clusters $C_1, \ldots, C_L$ and final vectors $\mathbf{c}_1, ..., \mathbf{c}_L$.

Figure 3.30: Basic leader-follower clustering algorithm (single-pass)

Basically, the leader-follower clustering algorithm is a typical single-pass method for DC. Documents are sequentially checked from the top to bottom of the target file, in which clusters are generated and updated incrementally. Thus the number of comparisons between documents and clusters is not over $N \times L$ times, and it is relatively easy to apply the algorithm to large-scale sets of documents.

However, the number of comparisons approaches to $O(N^2)$ if L becomes larger and $L \approx N$. Also, like incremental k-means algorithms with no iteration, clustering results are highly dependent on the order of documents to be processed. Table 3.14 shows two results of leader-follower clustering for the sample DB in Table 3.1. One result was obtained by processing documents in ascending order from d_1 to d_{10}, and the other was in descending order from d_{10} to d_1.

In the execution of leader-follower clustering shown in Table 3.14, tf (term frequency) was simply used as w_{ij} in each document vector, and $\tilde{w}_{kj}$ of cluster vectors was computed such that

$$\tilde{w}_{kj} = \sum_{i:d_i \in C_k} \mathrm{f}_{ij}, \tag{3.56}$$

which means that each cluster is regarded as a huge document created by concatenating documents belonging to the cluster. When tf vector is adopted, an element of the cluster centroid is calculated as $m_{kj} = \tilde{w}_{kj}/\tilde{n}_k = \sum_i \mathrm{f}_{ij}/\tilde{n}_k$. Results in Table 3.14 were obtained by using the cosine coefficient

Table 3.15: Result of leader-follower clustering for sample DB (2)

θ_s		Ascending order: $d_1 \to d_{10}$
0.2	$L = 2$	$\{d_1, d_2, d_3, d_5, d_6, d_8, d_9\}, \{d_4, d_7, d_{10}\}$
0.3	$L = 3$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
0.4	$L = 4$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}$
θ_s		Descending order: $d_{10} \to d_1$
0.2	$L = 2$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_6, d_7, d_8, d_9, d_{10}\}$
0.3	$L = 3$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}$
0.4	$L = 4$	$\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}$

Note: Double-pass clustering.

as similarity measure $s(\mathbf{d}_i, \mathbf{c}_k)$, in which elements $\tilde{w}_{kj}$ and $\tilde{w}_{kj}/\tilde{n}_k$ provide the same result [41].

It is clear from Table 3.14 that the two results are different, and the result by processing in descending order appears to be slightly more valid. For example, in comparison with dendrograms in Figure 3.5(b) and Figure 3.7(a) and results from the k-means algorithm in Table 3.4, the partition, $C_1 = \{d_1, d_2, d_3, d_5\}$, $C_2 = \{d_4, d_7\}$ and $C_3 = \{d_6, d_8, d_9, d_{10}\}$, in the case of ascending order with $\theta_s = 0.3$ may be slightly problematic whereas the descending order with $\theta_s = 0.3$ provides the same partition with that by the k-means algorithm [42].

A method for reducing the degree of dependency on the order of documents would be *double-pass clustering* [71], in which the document file is scanned twice. In the first scan, a standard leader-follower clustering algorithm is executed, and in the second scan, all documents are reallocated to the most similar cluster without updating cluster vectors (typically, similarity threshold θ_s is not applied to the reallocation in the second scan for exclusive clustering). Results of double-pass leader follower clustering is shown in Table 3.15.

Double-pass clustering in two cases of $\theta_s = 0.3$ and $\theta_s = 0.4$ yielded the same partition both in ascending and descending orders, which also equals to those by the k-means algorithm in Table 3.4. This example implies that influence of the order is successfully weakened in double-pass clustering. Unfortunately, when $\theta_s = 0.2$, it was not the case (but dividing naturally the sample DB into two separate parts may be difficult as shown in dendrograms in Figure 3.5(b) and Figure 3.7(a)). Although processing time increases approximately twice, it would be worth while attempting the double-pass clustering for obtaining better results.

In order to avoid double increase of the processing time, it may be possible to stop the first scan at n-th document from the top of the target file (where $n < N$) [43]. For example, when $N = 10000$ and $n = 1000$, the number of comparisons becomes $1,000 + 10,000 = 11,000$, which is only 110 %, not 200 %, of single-pass clustering. If cluster vectors having enough representativeness are constructed by checking only the n documents, then quality of the clustering result would not deteriorate largely. Of course, it is highly dependent on a condition of target document files. When documents on various topics are randomly distributed in the entire file, the reduction of documents checked in the first scan would not decrease cluster validity largely. In contrast, if documents on particular topics are placed in top position of the file and the others appear in bottom parts, then quality of the clustering result becomes inevitably low [44].

---

[41] Since

$$s(\mathbf{d}_i, \mathbf{c}_k) = \frac{\sum_{j=1}^{M} w_{ij}\tilde{w}_{kj}/\tilde{n}_k}{\sqrt{\sum_{j=1}^{M} w_{ij}^2}\sqrt{\sum_{j=1}^{M}(\tilde{w}_{kj}/\tilde{n}_k)^2}} = \frac{\tilde{n}_k^{-1}\sum_{j=1}^{M} w_{ij}\tilde{w}_{kj}}{\sqrt{\tilde{n}_k^{-2}}\sqrt{\sum_{j=1}^{M} w_{ij}^2}\sqrt{\sum_{j=1}^{M}\tilde{w}_{kj}}},$$

$\tilde{n}_k^{-1}$ is canceled out during computation of the cosine coefficient.

[42] Actually, it would be difficult to determine a cluster to which t_5 belongs in the sample DB. This will become clearer in experiments of probabilistic and matrix-based clustering in Chapter 4. Therefore, it would be impossible to conclude that the partition in ascending order with $L = 3$ is invalid.

[43] This idea was suggested by Kishida(2010) [152].

[44] Rather, in an experiment reported by Kishida(2010) [152] using RCV1 [187] as a test set, the reduction of documents checked in the first scan enhanced the effectiveness of clustering under a condition. Note that this experiment used logarithm-based tf-idf weighting in document vectors,

$$w_{ij} = (\log \mathrm{f}_{ij} + 1.0) \times \log \frac{N}{n_j}, \quad i = 1, \ldots, N;\ j = 1, \ldots, M,$$

3.3.3 Cover-coefficient-based concept clustering

As a predefined parameter, the leader-follower clustering algorithm requires similarity threshold θ_s, and the number of clusters L has to be given for execution of k-means clustering. However, in many situations of DC, it would be difficult to specify correctly an optimal value of θ_s or L.

C^{3}M (cover-coefficient-based concept clustering methodology) [51, 52, 53, 54] can work without such kind of predetermined parameters by predicting L from 'cover coefficient', which measures the degree to which a given document is 'covered' by the other documents. In computing the cover coefficient, two quantities ϕ_{ij} and $\tilde{\phi}_{ij}$ are first defined such that

$$\phi_{ij} = \frac{b_{ij}}{\sum_{j=1}^{M} b_{ij}},\ i = 1, ..., N; j = 1, ..., M, \tag{3.57}$$

$$\tilde{\phi}_{ij} = \frac{b_{ij}}{\sum_{i=1}^{N} b_{ij}},\ i = 1, ..., N; j = 1, ..., M, \tag{3.58}$$

where if $\mathrm{f}_{ij} \geq 1$, then $b_{ij} = 1$, and otherwise, $b_{ij} = 0$. For example, in the sample DB in Table 3.1, $\phi_{12} = 1/2$ because distinct two terms appears in document d_1, and $\phi_{72} = 1/4$, which suggests that 'importance' of t_2 is higher in d_1 than in d_7. Also, $\tilde{\phi}_{11} = 1/3$ because t_1 is contained in three documents, and thus $\tilde{\phi}_{ij}$ corresponds to inverse document frequency of t_j.

The 'cover coefficient' is calculated such that

$$\delta_{ii'} = \sum_{j=1}^{M} \phi_{ij}\tilde{\phi}_{i'j}, \tag{3.59}$$

based on the two quantities, ϕ_{ij} and $\tilde{\phi}_{ij}$. In the case that $i = i'$, δ_{ii} can be considered to measure 'uniqueness' of document d_i, and its value becomes larger when the degree of uniqueness is higher. Actually, if document d_i does not share any term with the other documents, then $\delta_{ii} = 1$, which means that d_i is completely unique in the document set. Conversely, when all terms in document d_i appear in all the other documents, δ_{ii} becomes $1/N$, which is the minimum value of δ_{ii}.

For example, in the sample DB,

$$\delta_{11} = \phi_{11} \times \tilde{\phi}_{11} + \phi_{12} \times \tilde{\phi}_{12} = \frac{1}{2} \times \frac{1}{3} + \frac{1}{2} \times \frac{1}{6} = 0.250, \tag{3.60}$$

and

$$\delta_{88} = \phi_{84} \times \tilde{\phi}_{84} + \phi_{85} \times \tilde{\phi}_{85} = \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times \frac{1}{6} = 0.208, \tag{3.61}$$

from which it turns out that d_1 is more 'unique' than d_8. The difference is caused from the fact that the number of documents including t_1, which is a factor of δ_{11}, is smaller than that of t_4 in δ_{88} (the numbers of document including t_2 and t_5 are equal).

When all documents do not share any terms with the other documents (i.e., $\delta_{ii} = 1$ for $i = 1, \ldots, N$), any merging of documents should not be done, and inevitably, $L = N = \sum_{i=1}^{N} \delta_{ii}$. In contrast, if all documents have the same terms, then they may have to be merged into a cluster (i.e., $L = 1$). In this case, $\sum_{i=1}^{N} \delta_{ii} = \sum_{i=1}^{N} 1/N = 1$. The two extreme examples lead to estimating the number of clusters by

$$\hat{L} = \sum_{i=1}^{N} \delta_{ii} = \sum_{i=1}^{N} \sum_{j=1}^{M} \frac{b_{ij}}{\sum_{j'=1}^{M} b_{ij'}} \frac{1}{n_j}. \tag{3.62}$$

For example, in the sample DB, $\hat{L} = \sum_{i=1}^{N} \delta_{ii} = 2.381$, which suggests that $L = 2$ or $L = 3$ should be selected.

---

and that elements of k-th cluster vector were defined by

$$\tilde{w}_{kj} = \log \sum_{i:d_i \in C_k} \mathrm{f}_{ij} + 1.0., \quad j = 1, \ldots, M$$

(see Section 5.3.3 for details).

Suppose that there is a document set whose matrix of tf is shown in Figure 3.31(1). Clearly, $L = 3$ because three subsets of documents, $\{d_A, d_B\}$, $\{d_C, d_D, d_E\}$ and $\{d_F\}$, do not have any common term and are completely exclusive. Within a subset of the example (1) in Figure 3.31,

$$\sum_{i \in C_k} \delta_{ii} = \tilde{n}_k \sum_{j=1}^{M} \frac{b_{ij}}{\sum_{j'=1}^{M} b_{ij'}} \frac{1}{n_j} = \frac{\tilde{n}_k}{n_j} \sum_{j=1}^{M} \frac{b_{ij}}{\sum_{j'=1}^{M} b_{ij'}} = 1 \times 1 = 1, \tag{3.63}$$

because n_j is a constant for all terms appearing in the subset C_k, and equals to $\tilde{n}_k$ in this special situation. Hence, it can be obtained exactly that $\hat{L} = 3$ in Equation (3.62) for this example [45].

	dA	dB	dC	dD	dE	dF
t1	2	1	0	0	0	0
t2	1	1	0	0	0	0
t3	0	0	3	1	1	0
t4	0	0	1	2	1	0
t5	0	0	0	0	0	2

Example (1)

	dA	dB	dC	dD	dE	dF
t1	2	1	0	0	0	1
t2	1	1	1	0	0	0
t3	1	0	3	1	1	0
t4	0	0	1	2	1	0
t5	0	0	0	0	0	2

Example (2)

	dA	dB	dC	dD	dE	dF
t1	2	1	0	0	0	0
t2	0	1	0	0	0	0
t3	0	0	3	1	1	0
t4	0	0	0	2	1	0
t5	0	0	0	0	0	2

Example (3)

Figure 3.31: Examples of document set (term-by-document matrices)

In example (2) of Figure 3.31, d_A, d_C and d_F have a term appearing in a subset other than the own subset. In this case, $\hat{L}$ is smaller (i.e., $\hat{L} = 2.19$). Also, in example (3) of Figure 3.31, 'uniqueness' of d_A and d_C is higher, which makes value of $\hat{L}$ larger (i.e., $\hat{L} = 3.41$).

Basically, the C^{3}M is a variation of k-medoids clustering, in which a typical document is used for representative of a particular cluster. Such documents are selected based on 'cluster seed power' that is defined as

$$\kappa_i = \delta_{ii}(1 - \delta_{ii}) \sum_{j=1}^{M} b_{ij}, \tag{3.64}$$

where $\sum_{j=1}^{M} b_{ij}$ corresponds to the number of distinct terms appearing in the document. Note that the maximum value of $\delta_{ii}(1 - \delta_{ii})$ is 0.25 when $\delta_{ii} = 0.5$. In the sample DB, d_7 has the maximum 'cluster seed power' (i.e, $\kappa_7 = 0.789$) which is followed by d_{10} ($\kappa_{10} = 0.636$).

A seed document selected from values of κ_i in Equation (3.64) is denoted by $d_{i'}$ here. In the C^{3}M, document d_i other than seed documents is assigned to seed document $d_{i'}$ whose $\delta_{ii'}$ is the maximum. In the sample DB, if it is assumed that d_7 and d_{10} are first and second seeds, respectively (i.e., $L = 2$), then $C_1 = \{d_1, d_2, d_3, d_4, d_5, d_6, d_7\}$ and $C_2 = \{d_8, d_9, d_{10}\}$ are generated. For example, $\delta_{57} = 0.194$ and $\delta_{5,10} = 0.055$. In the case of d_4, these values are equal, but cluster of d_7 is selected because d_7 has greater 'cluster seed power' than d_{10} [46]. Note that $\delta_{ii'}$ tends to become greater as the two documents share many more terms and the common terms have larger idf factor (see Equation (3.59)).

[45] Of course, this result is easily conjectured from the fact that $\sum_{i=1}^{N} \delta_{ii} = 1$ when all documents have the same terms in the entire set.

[46] Also, cover-coefficient-based incremental clustering methodology(C^{2}ICM) has been proposed for modifying a result by the C^{3}M (see [49, 50]).

3.3.4 Scatter/Gather algorithm

Scatter/Gather algorithm [72, 123, 284] allows users to obtain a small subset of documents relevant to their information needs from a large-scale document collection by repeating operations of 'scatter' and 'gather', not executing keyword searches. The 'scatter' tries to divide automatically a set of documents into subsets, and in the 'gather' stage, the user selects manually relevant subsets among them and a new target set is created by concatenating these subsets. Repeatedly, the same procedure will be applied to the new set, and gradually, relevant documents are filtered out into a smaller subset.

More precisely, the 'scatter' is executed as follows:

1. Find L centroids by using a hierarchical clustering algorithm.
2. Apply a k-means algorithm.
3. Refine clusters obtained in previous step.

In order to alleviate a problem of high computational complexity for executing hierarchical clustering in the first step, 'Buckshot' and 'Fractionation' algorithms were proposed [72].

In the Buckshot, $\sqrt{L \times n}$ documents are randomly selected from the set of documents, and hierarchical clustering is executed for the random sample, where n denotes the number of documents included in the current set (initially, $n = N$). Although an incorrect clustering result may be obtained due to using only a part of the set, the number of document pairs reduces to $O(Ln)$ from $O(n^2)$. Because the purpose of hierarchical clustering in this case is to find cluster centroids for the k-means algorithm, it is enough that the random sample represents an unbiased 'picture' of the entire set to a degree. Also, from the same reason, the hierarchical clustering is stopped when L clusters are obtained.

In contrast, the Fractionation algorithm divides automatically the entire set into some smaller subsets, and a hierarchical clustering algorithm is applied to each subset independently. In next step, by regarding each resulting cluster as a single 'huge' document, the same procedure is repeated. Although the Fractionation would take much more time than the Buckshot, it may be possible to provide better results because all documents in the set enter into the process.

Suppose that document set D is simply partitioned into N/m parts, each of which includes m documents, respectively. If $\rho \times m$ clusters are generated for each part ($0 < \rho < 1$) by applying a hierarchical clustering algorithm, then in total, the number of clusters amounts to $N/m \times \rho m = \rho N$. These clusters are denoted by $C_{k,h}$ ($k = 1, ..., N/m$; $h = 1, ..., \rho m$) here. By considering each $C_{k,h}$ as a single 'huge' document, the ρN pseudo-documents are divided into m parts again and the same procedure is applied. As a result, $\rho^2 N$ clusters are obtained because $\rho N/m \times \rho m = \rho^2 N$. After this procedure is repeated r times, the current clusters are adopted as seeds for k-means clustering in the next step if $\rho^r N \leq L$. Its computational complexity becomes $O(m^2 \times N/m + m^2 \times \rho N/m + ...) = O(Nm(1 + \rho + \rho^2 + ... + \rho^r)) = O(Nm)$ because $0 < \rho < 1$ [47].

Table 3.16 shows values of $\rho^r N$ when N =10000, 50000 and 100000, and $\rho = 0.2$ and 0.5. In the case that $m = 10$, $N = 10000$ and $\rho = 0.2$, and that the iterative process is stopped at $r = 4$, it amounts to

$$Nm(1 + \rho + \rho^2 + ... + \rho^r) = 10000 \times 10 \times (1 + 0.2^1 + \ldots + 0.2^4) = 124964.$$

After the computation, cluster centroids for $L = 16$ can be obtained (see Table 3.16). It should be noted that ten documents are grouped into 2 ($= 10 \times 0.2$) clusters at each iteration in this case. For obtaining valid clusters, it is desirable that each set of 10 documents is topically homogeneous to a degree. If the topical homogeneity can not be assumed, then larger m should be adopted. For example, when $m = 100$, 100 documents are reduced into 20 pseudo documents at each iteration. Inevitably, the value of Nm becomes large, which leads to higher computational complexity.

Finally, in order to refine a result of k-means clustering, each cluster is divided by using the Backshot and two clusters sharing the same terms are joined [72].

[47] Actually, $\sum_{s=0}^{r} \rho^s = (1 - \rho^{r+1})/(1 - \rho)$ (i.e. the sum of the terms of a geometric progression), which converges to a constant $(1 - \rho)^{-1}$ as $r \to \infty$ if $0 < \rho < 1$.

Table 3.16: Values of $\rho^r N$ at each iteration of Fractionation

r	$\rho = 0.2$			$\rho = 0.5$		
	$N = 10000$	50000	100000	$N = 10000$	50000	100000
1	2000.00	10000.00	20000.00	5000.00	25000.00	50000.00
2	400.00	2000.00	4000.00	2500.00	12500.00	25000.00
3	80.00	400.00	800.00	1250.00	6250.00	12500.00
4	16.00	80.00	160.00	625.00	3125.00	6250.00
5	3.20	16.00	32.00	312.50	1562.50	3125.00
6	0.64	3.20	6.40	156.25	781.25	1562.50
7		0.64	1.28	78.13	390.63	781.25
8			0.26	39.06	195.31	390.63
9				19.53	97.66	195.31
10				9.77	48.83	97.66

3.3.5 Self-organizing map

Clustering based on SOM

Self-organizing map (SOM) by Kohonen [165] is widely used for visualizing a latent structure in various fields, and is often applied to DC (e.g., [190, 59, 189, 229, 262, 35]). The research group of Kohonen has also explored a method of DC based on SOM, which is specially called WEBSOM [131].

The SOM is a kind of *neural network* that is composed of a pair of input space and output layer. In the case of DC, $\mathbf{d}_i$ is used as a vector of the input space, and a 'node' in the output layer corresponds to a cluster of documents. For example, if the output layer is formalized as a rectangular consisting of 5 cells on x-axis and 10 cells on y-axis, then $5 \times 10 = 50$ nodes are established in the output layer. In this section, a feature vector representing each node in the output is denoted by $\mathbf{v}_h$ $(h = 1, ..., H)$ where H means the total number of nodes.

The SOM consists of 'competitive' stage and 'collaborative' stage, and the two stages are iterated for developing nodes in the output layer. If similarity between input vector $\mathbf{d}_i$ and h-th output vector $\mathbf{v}_h$ $(i = 1, \ldots, N; h = 1, \ldots, H)$ is measured by squared Euclidean distance [229], then the nearest node of input vector $\mathbf{d}_i$ can be determined as

$$h^* = \arg\min_h \|\mathbf{d}_i - \mathbf{v}_h\|^2 = \arg\min_h \sum_{j=1}^{M} (w_{ij} - v_{hj})^2, \tag{3.65}$$

where $\mathbf{v}_h = [v_{h1}, ..., v_{hM}]^T$, and it is feasible to interpret the h^*-th node as a 'winner'. The purpose of competitive stage is to select the winner for each input vector.

The collaborative stage tries to adjust weights in vectors of the 'winner' node and its neighbors. Let $\mathcal{N}_{h^*}$ denote the set of the 'winner' node for d_i and its neighbors. The nodes in $\mathcal{N}_{h^*}$ can be updated such that

$$v_{hj}^{(s+1)} = \begin{cases} v_{hj}^{(s)} + \eta^{(s)}(w_{ij} - v_{hj}^{(s)}), & \text{if } h\text{-th node is included in } \mathcal{N}_{h^*} \\ v_{hj}^{(s)}, & \text{otherwise} \end{cases}, \tag{3.66}$$

where s means current number of iterations and $\eta^{(s)}$ indicates 'learning rate' which decreases as s increases [229].

Figure 3.32 shows a typical DC algorithm based on the SOM, and Figure 3.33 indicates an example of SOM-based clustering for the sample DB, which was computed from unit vectors $\mathbf{d}_i/\|\mathbf{d}_i\|$ $(i = 1, \ldots, N)$ by using somgrid() and SOM() of R [246]. In Figure 3.33, whereas H was set to six a priori $(= 2 \times 3)$, resulting feature vectors of three cells became identical (the cells include only d_7 in the figure), and therefore, the number of clusters is four (i.e., $L = 4$). Labels representing a topic of each cell can be automatically assigned based on term frequencies in documents belonging to the cell [171].

‡ Simple SOM-based document clustering

Set: The number of nodes in output layer H and the number of iterations r.

1) Initialize output vector $\mathbf{v}_h$ $(h = 1, ..., H)$ by generating random numbers.

2) For d_i $(i = 1, \ldots, N)$, a winner is determined by Equation (3.65) and feature vectors of neighbor nodes are adjusted by Equation (3.66). This procedure is repeated r times.

3) Each document d_i is allocated to the nearest node $(i = 1, \ldots, N)$.

Out: The number of clusters L, clusters $C_1, \ldots, C_L$ and final vectors of nodes in output layer $\mathbf{v}_1, ..., \mathbf{v}_H$.

Figure 3.32: Simple SOM-based document clustering

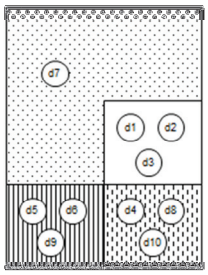


Figure 3.33: Example of SOM ($2 \times 3 = 6$ units)

WEBSOM

The WEBSOM is tailored so as to be applied for partitioning a large-scale document set. For example, over six million abstracts of patent materials were clustered by the WEBSOM [166].

In the WEBSOM, feature vectors in the output layer are modified such that

$$v_{hj}^{(s+1)} = v_{hj}^{(s)} + f_{c(\mathbf{d}_i),h}(s)(w_{ij} - v_{hj}^{(s)}), \tag{3.67}$$

where $c(\mathbf{d}_i)$ is a function returning an index number of the winner node for d_i, and

$$f_{c(\mathbf{d}_i),h}(s) = \alpha(s) \exp\left(-\frac{\|\mathbf{t}_h - \mathbf{t}_{c(\mathbf{d}_i)}\|^2}{2\sigma^2(s)}\right). \tag{3.68}$$

In the equation, $\mathbf{t}_h$ denotes two dimensional vectors indicating a location of h-th node, $\alpha(s)$ indicates learning rate in s-th iteration, and $\sigma^2(s)$ is a parameter for adjusting variance of the function $f_{c(\mathbf{d}_i),h}(s)$. Due to using the function, it is not necessary to define explicitly a set of neighbors $\mathcal{N}_{h^*}$.

Also, in order to execute the WEBSOM for large-scale document sets, dimensional reduction of document vectors has been attempted by using a conversion in Equation (3.12) as described above. Although it is possible to use LSI technique (Section 4.2.1) for defining $\mathbf{B}$, the matrix may be randomly generated so that elements of each column are normally distributed [141, 166].

3.3.6 Fuzzy clustering

Fuzzy c-means algorithm

The degree to which a data point (e.g., a document) is related with a cluster can be interpreted as a value of membership function in a 'fuzzy set' developed by L. A. Zadeh, which naturally leads to

an idea of *fuzzy clustering* [232]. A basic method of the fuzzy clustering would be *fuzzy c-means* (FCM) algorithm [86, 25]. This algorithm is a variation of k-means, and tries to maximize an objective function,

$$Q = \sum_{k=1}^{L} \sum_{i=1}^{N} u_{ki}^{m} \, \|\mathbf{d}_i - \mathbf{c}_k\|^2, \tag{3.69}$$

where $\mathbf{c}_k$ denotes k-th cluster vector, $\mathbf{U} = [u_{ki}]$ indicates a 'partition matrix', and m means the 'fuzzy factor' here. In DC, the partition matrix is estimated from data $\mathbf{d}_1, \ldots, \mathbf{d}_N$ under a condition that $\sum_k u_{ki} = 1$. Thus an element of $\mathbf{U}$ looks like formally 'probability' (i.e., $0 \leq u_{ki} \leq 1$) representing the degree to which a document belongs to a cluster. In addition, the 'shape' of each cluster can be changed by using different fuzzy factor m (see [232] for details) [48].

By adding the Lagrange multiplier λ into objective function Q for the condition $\sum_k u_{ki} = 1$, and differentiating it by u_{ki}, an equation for estimating u_{ki} becomes that $m u_{ki}^{m-1} \|\mathbf{d}_i - \mathbf{c}_k\|^2 - \lambda = 0$, which leads to

$$u_{ki} = \left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} \frac{1}{\|\mathbf{d}_i - \mathbf{c}_k\|^{\frac{2}{m-1}}}. \tag{3.70}$$

Since $\sum_k u_{ki} = 1$,

$$\left(\frac{\lambda}{m}\right)^{\frac{1}{m-1}} = \left(\sum_{k'=1}^{L} \frac{1}{\|\mathbf{d}_i - \mathbf{c}_{k'}\|^{\frac{2}{m-1}}}\right)^{-1}, \tag{3.71}$$

is obtained, and therefore, u_{ki} can be estimated as

$$u_{ki} = \left[\sum_{k'=1}^{L} \left(\frac{\|\mathbf{d}_i - \mathbf{c}_k\|}{\|\mathbf{d}_i - \mathbf{c}_{k'}\|}\right)^{\frac{2}{m-1}}\right]^{-1} \tag{3.72}$$

$(k = 1, \ldots, L;\ i = 1, \ldots, N)$.

Also, since $\partial Q / \partial \mathbf{c}_k = -2 \sum_i u_{ki}^m (\mathbf{d}_i - \mathbf{c}_k)$, equation $\partial Q / \partial \mathbf{c}_k = \mathbf{0}$ leads to $\mathbf{c}_k = \sum_i u_{ki}^m \mathbf{d}_i / \sum_i u_{ki}^m$ [49]. Thus it is feasible to estimate u_{ki} $(k = 1, \ldots, L;\ i = 1, \ldots, N)$ and $\mathbf{c}_k$ $(k = 1, \ldots, L)$ by iterative updates of $\mathbf{c}_k$ by this equation and of u_{ki} by Equation (3.72), respectively, after initialization of the partition matrix by generating random numbers. For example, the algorithm created a partition matrix such that

$$\mathbf{U} = \begin{bmatrix} 0.50 & 0.98 & 0.94 & 0.09 & 0.30 & 0.07 & 0.07 & 0.05 & 0.02 & 0.24 \\ 0.26 & 0.01 & 0.03 & 0.79 & 0.53 & 0.09 & 0.87 & 0.07 & 0.02 & 0.37 \\ 0.23 & 0.01 & 0.02 & 0.12 & 0.17 & 0.84 & 0.07 & 0.88 & 0.96 & 0.39 \end{bmatrix},$$

for the sample DB when $L = 3$ and $m = 2.0$ [50]. The partition matrix suggests to group documents such as $\{d_1, d_2, d_3\}$, $\{d_4, d_5, d_7\}$, and $\{d_6, d_8, d_9, d_{10}\}$, which is a 'valid' set of clusters. Cluster vectors were calculated such that

$$\begin{aligned} \mathbf{c}_1 &= [0.306, 0.842, 0.031, 0.018, 0.021, 0.021]^T, \\ \mathbf{c}_2 &= [0.036, 0.185, 0.743, 0.244, 0.147, 0.133]^T, \\ \mathbf{c}_3 &= [0.020, 0.064, 0.064, 0.267, 0.854, 0.044]^T. \end{aligned}$$

As suggested by the partition matrix, the degrees to which d_1, d_5 and d_{10} belong to clusters were relatively low (i.e., < 0.6). This may represent 'inter-disciplinary' nature of the three documents, or they may be 'heterogeneous' in the document set. Suppose that external knowledge on heterogeneity of d_1, d_5 and d_{10} in the document set was given before executing DC. The knowledge can be incorporated into fuzzy clustering by assuming that

$$\sum_{k=1}^{L} u_{ki} = f_i, \tag{3.73}$$

[48] Furthermore, by adopting other types of distance (e.g., the Hamming distance, Tchebyschev distance, Minkowski distance and so on) in Equation (3.69), the shape of clusters is drastically changed [232].

[49] Differentiation of a function with respect to a vector is defined as $\partial f(\mathbf{x}) / \partial \mathbf{x} = [\partial f(\mathbf{x}) / \partial x_1, \ldots, \partial f(\mathbf{x}) / \partial x_n]^T$ where $\mathbf{x} = [x_1, \ldots, x_n]^T$ and $f : \mathbb{R}^n \to \mathbb{R}$. When $f(\mathbf{x}) = \mathbf{x}^T \mathbf{y}$, $\partial f(\mathbf{x}) / \partial \mathbf{x} = \mathbf{y}$ for two vectors $\mathbf{x}$ and $\mathbf{y}$.

[50] In this run, the matrix converged after 23 iterations. Of course, the number of iterations to be required depends on the initial matrix and the criterion of converge.

where, for example, $f_1 = f_5 = f_{10} = 0.5$ and $f_i = 1.0$ for the other documents. Equation (3.73) can be considered to insert a 'condition' or 'context' into fuzzy clustering, which is often called *conditional fuzzy clustering* [232]. In this case, Equation (3.72) changes to

$$u_{ki} = f_i \left[\sum_{k'=1}^{L} \left(\frac{\|\mathbf{d}_i - \mathbf{c}_k\|}{\|\mathbf{d}_i - \mathbf{c}_{k'}\|} \right)^{\frac{2}{m-1}} \right]^{-1}, \tag{3.74}$$

according to the condition of Equation (3.73) [51].

For the sample DB, the conditional fuzzy clustering generated a partition matrix such that

$$\mathbf{U} = \begin{bmatrix} 0.24 & 1.00 & 0.99 & 0.06 & 0.17 & 0.06 & 0.07 & 0.05 & 0.01 & 0.123 \\ 0.14 & 0.00 & 0.01 & 0.86 & 0.23 & 0.07 & 0.86 & 0.07 & 0.01 & 0.186 \\ 0.13 & 0.00 & 0.01 & 0.08 & 0.10 & 0.87 & 0.07 & 0.88 & 0.98 & 0.193 \end{bmatrix},$$

from which the same set of clusters is suggested. In contrast, the cluster vectors moved to

$$\begin{aligned} \mathbf{c}_1 &= [0.254, 0.931, 0.013, 0.006, 0.009, 0.006]^T, \\ \mathbf{c}_2 &= [0.011, 0.114, 0.832, 0.292, 0.109, 0.104]^T, \\ \mathbf{c}_3 &= [0.006, 0.062, 0.063, 0.250, 0.907, 0.012]^T, \end{aligned}$$

respectively. Because effect of d_1, d_5 and d_{10} located in middle points among three clusters decreases by introducing the condition f_i, the three clusters become more apart from each other. Actually, whereas the average Euclidean distance between three clusters is 1.16 in the standard fuzzy clustering, that in the conditional fuzzy clustering amounts to 1.38.

Fuzzy co-clustering

Like the SKWIC algorithm (Figure 3.29), the fuzzy clustering algorithm can be applied for grouping documents and terms simultaneously, which is called *fuzzy co-clustering* [169]. For instance, the objective function becomes

$$Q_c = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ki} v_{kj} w_{ij}, \tag{3.75}$$

where v_{kj} indicates the degree to which cluster C_k belongs to term t_j (i.e., $\sum_j v_{kj} = 1$). By maximizing the objective function under some assumptions and constrains, u_{ki} and v_{kj} can be estimated for fuzzy co-clustering [52].

3.3.7 Text stream clustering

In some applications of DC, it may be better to regard the target set explicitly as a 'text stream' in which documents are chronologically ordered. For instance, a system of detecting automatically current news topics would have to process a stream of news articles arriving constantly through the Internet, and to group them effectively and efficiently. Such kind of *text stream clustering* is a relatively new research topic of DC.

---

[51] The function for minimizing Q becomes

$$\mathcal{H}_i = \sum_k u_{ki}^m \|\mathbf{d}_i - \mathbf{c}_k\|^2 - \lambda \left(\sum_k u_{ki} - f_i \right),$$

for each d_i $(i = 1, \ldots, N)$.

[52] For instance, in [169],

$$\mathcal{H} = \sum_{i=1}^{N} \sum_{j=1}^{M} u_{ki} v_{kj} w_{ij} - T_u \sum_{k=1}^{L} \sum_{i=1}^{N} u_{ki}^2 - T_v \sum_{k=1}^{L} \sum_{j=1}^{M} v_{kj}^2 + \sum_{i=1}^{N} \lambda_i \left(\sum_{k=1}^{L} u_{ki} - 1 \right) + \sum_{k=1}^{L} \gamma_k \left(\sum_{j=1}^{M} v_{kj} - 1 \right),$$

is maximized by differentiating it where λ_i and γ_k are the Lagrange multipliers and T_u and T_v are parameters. In experiment reported by [169], $T_u = 0.00001$ and $T_v = 1.5$.

Online processing for text stream clustering

For the text stream clustering, it is possible to assume two different situations on availability of text data as follows:

- Full storage. Text data of all documents from the beginning of the stream can be fully stored in disk(s) (e.g., in the form of document vector), which are available for generating clusters at any time point.
- Online processing. Text data of each document are not stored and only representations of clusters (e.g., cluster vectors) generated in past are available.

Usually, literature of text stream clustering appears to focus implicitly or explicitly on online situations. Inevitably, under the assumption, online or incremental clustering techniques (Section 3.3.1) for which iterative computation in batch mode is prohibitive have to be applied. In the case that the number of clusters is unknown, a natural way would be to use a single-pass leader-follower clustering algorithm (Section 3.3.2) for online text stream clustering. Actually, some researchers have adopted the algorithm or its variants in their experiments (e.g., [6, 327, 182, 326]). Of course, it may not be easy to determine the threshold value θ_s in assigning a document to a cluster.

If the number of clusters can be reasonably assumed a priori, then it is possible to apply online k-means algorithms. Especially, the simple scalable k-means algorithm [340] (see Section 3.3.1 for details) would be appropriate in this situation. In the case that a set of documents arriving at a particular time interval can be regarded as each sample, this algorithm would be available straightforwardly.

When computing term weights used in a clustering algorithm, it is necessary to select carefully a method for calculating the idf factor because the arrival of new documents changes the number of documents including a particular term. Possibly, another corpus in a similar domain would be required for obtaining 'valid' idf weights (e.g., $\log(N/n_j)$) at early stage of the stream if the idf factor must be included. In a continuing process of text stream clustering, whether the idf factor in term weighting is incrementally updated at arrival of new documents or not may be an important decision [327].

Incorporation of time-dependent factor

In the case of applying a clustering algorithm to chronologically ordered documents, it is natural to suppose that the contents of two documents published in different time periods tend to be dissimilar, which leads to incorporating a *time-decay factor* into computation of similarity measure between two documents, between two clusters, or between a document and a cluster [53].

For instance, if similarity measure between a document vector and a centroid vector is denoted by $s(\mathbf{d}_i, \mathbf{m}_k)$ where $\mathbf{m}_k$ is centroid of cluster C_k, then a discounted similarity incorporating a 'time penalty' factor written as $T(d_i, C_k)$ can be calculated such that

$$s'(d_i, C_k) = s(\mathbf{d}_i, \mathbf{m}_k) - T(d_i, C_k), \tag{3.76}$$

or

$$s'(d_i, C_k) = s(\mathbf{d}_i, \mathbf{m}_k)/T(d_i, C_k). \tag{3.77}$$

Actually, it is assumed that $T(d_i, C_k)$ increments as time interval between d_i and documents included in C_k becomes large. As exemplified in the equations, the similarity measure used in text stream clustering may consist of content-factor and time-factor. The actual form of $T(d_i, C_k)$ depends on the clustering technique to be used [54].

Additionally, identification of a 'bursty feature' may be helpful to obtain a good result of text stream clustering [121, 132]. For example, a word 'Olympic' would be suddenly referred to in so many documents at a specific period (i.e., when the Olympic Games are going on). Algorithms

[53] Note that, in some cases, after a set of new documents arriving in a given period were grouped into clusters, the new clusters are merged into 'old' clusters generated in the past according to a similarity measure (e.g., cosine coefficient of two centroid vectors).

[54] As to examples of time-decay factor for the leader-follower and spherical k-means algorithms, see [6] and [340], respectively.

for detecting automatically the sudden burst in words (or phrases) included in a text stream have been developed so far, and it is expected that DC quality is improved by augmenting weights of such words (i.e., 'bursty feature') in computation for clustering because the bursty feature possibly indicates a single event or phenomena in the burst period, and is not ambiguous within the period.

For identifying the bursty features from text stream data, Kleinberg's two-state finite automaton model [160] can be used. After that, the weight of each bursty feature in documents created in the burst period is augmented such that $w'_{ij} = w_{ij} + \delta b_j$ where b_j is 'bursty weight' and δ indicates a coefficient. The bursty weight can be computed from the Kleinberg's model, and δ has to be set as a positive constant.

Topic detection

Research efforts on text stream clustering were partially included in the *Topic Detection and Tracking* (TDT) research program since around 1997, which had been supported by the U.S. Government. An initial motivation of TDT research was to construct a system that "monitor broadcast news and alert an analyst to new and interesting events happening in the world" (p.1 in [5]). For developing component technologies of the system, some research institutes participated in the program and tackled the following five tasks[5]:

1. Story segmentation. Dividing the transcript of a news show into individual stories.
2. First story detection. Recognizing the onset of a new topic in the stream of news stories.
3. Cluster detection. Grouping all stories as they arrive, based on the topics they discuss.
4. Tracking. Monitoring the stream of news stories to find additional stories on a topic that was identified using several sample stories.
5. Story link detection. Deciding whether two randomly selected stories discuss the same news topic or not.

Among them, whereas machine learning approach can be applied to the tracking task, the cluster detection task is directly related with text stream clustering in unsupervised manner [55]. For instance, if the cluster detection is executed in 'online' mode, then a single-pass clustering algorithm incorporating the time-decay factor can be used (e.g., see [327]).

Also, such kind of single-pass clustering may be used for the 'first story detection' task in online mode, which is also called *novelty detection* in general [56]. A simple way is to consider that a document describes a 'new topic' if all similarities between the document and clusters created previously do not exceed a predetermined threshold. However, it may be better to compare similarities between the new document with 'old' documents arrived in a time window, respectively, without any clustering procedure [327]. For example, if $s(d_A, d_i) < \theta_s$ for all old documents d_i $(i = 1, ..., N)$ where θ_s is a threshold, then new arriving document d_A would be judged to contain a new topic not yet appearing in the set of $\{d_1, \ldots, d_N\}$.

Compact storage of cluster profiles

Another technical issue of text streaming clustering is how to create profiles of current clusters that exist at that time. It is hopeful that such kind of *cluster profile* keeps 'efficiently' information on the cluster, and enables to allocate 'effectively' new arriving documents to proper clusters (see [195] and [3] for detailed discussions).

[55] Due to the goal of TDT, 'topic' was operationally defined as "an event or activity, along with all directly related events and activities" [66]. Furthermore, the 'event' was also carefully defined in TDT.

[56] According to [206], novelty detection is "the identification of new or unknown data or signal that a machine learning system is not aware of during training".

Chapter 4

Probabilistic and Matrix-based Document Clustering

4.1 Probabilistic Document Clustering

Since early 2000s, advanced methods of probabilistic clustering for textual data have been developed in order to overcome problems in text mining. Although they provide usually a flat partition of a document set like k-means algorithms, underlying theories are complicated mathematically. This section reviews some of the methods with checking clustering results by them for the sample DB in Table 3.1, which would provide deeper insights about the mechanism of probabilistic clustering.

4.1.1 Clustering by mixture model

Poisson mixture model

A probability that document vector $\mathbf{d}_i$ is generated within cluster C_k is denoted by $P(\mathbf{d}_i|C_k)$ $(i = 1, \ldots, N; k = 1, \ldots, L)$ here. It is possible to partition a document set based on a probabilistic model of $P(\mathbf{d}_i|C_k)$. For instance, according to classical IR studies, *Poisson distribution* (Section 2.1.2) can be selected as the model. In this case, a probability of f_{ij}, which denotes occurrence frequency of term t_j (i.e., tf) in d_i as before, is written as

$$P(\mathrm{f}_{ij}|C_k) = \frac{\lambda_{j|k}^{\mathrm{f}_{ij}} e^{-\lambda_{j|k}}}{\mathrm{f}_{ij}!},\ \mathrm{f}_{ij} = 0, 1, 2, \ldots, \tag{4.1}$$

for cluster C_k where $\lambda_{j|k}$ is a parameter of the Poisson distribution for t_j in C_k. By assuming 'term independence' used often in IR studies [257, 253] (Section 2.1.4), $P(\mathbf{d}_i|C_k)$ can be computed as a simple multiplication of $P(\mathrm{f}_{ij}|C_k)$,

$$P(\mathbf{d}_i|C_k) = \prod_{j=1}^{M} P(\mathrm{f}_{ij}|C_k) = \prod_{j=1}^{M} \frac{\lambda_{j|k}^{\mathrm{f}_{ij}} e^{-\lambda_{j|k}}}{\mathrm{f}_{ij}!} \tag{4.2}$$

(note that $\mathbf{d}_i = [\mathrm{f}_{i1}, \ldots, \mathrm{f}_{iM}]^T$ here, i.e., which is a tf vector).

By using $P(\mathbf{d}_i|C_k)$ $(k = 1, \ldots, L)$, a *mixture model* [209] is constructed such as

$$P(\mathbf{d}_i) = \sum_{k=1}^{L} \eta_k P(\mathbf{d}_i|C_k), \tag{4.3}$$

where η_k is a mixing parameter, and since $0 \leq \eta_k \leq 1$, it is necessary that

$$\eta_L = 1 - \sum_{k=1}^{L-1} \eta_k. \tag{4.4}$$

Figure 4.1 shows two Poisson distributions of a single term in two clusters C_1 and C_2, respectively, and a mixture of them (*Poisson mixture*) with $\eta_1 = \eta_2 = 0.5$.

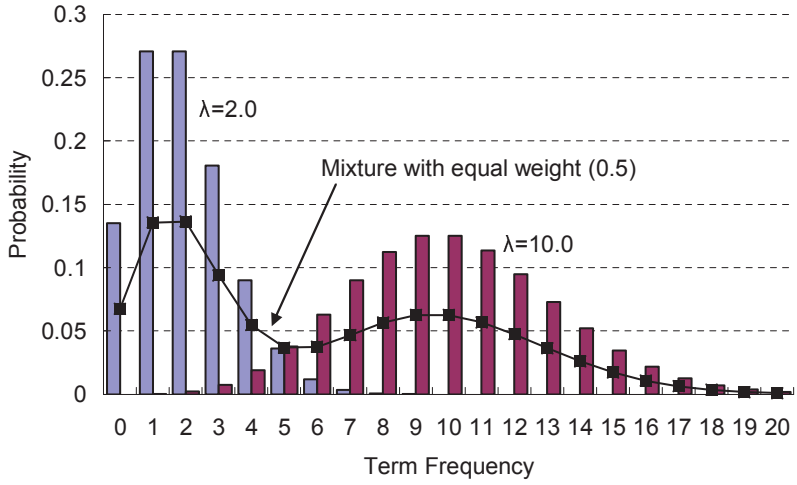


Figure 4.1: Two Poisson distributions and their mixture with $\eta_1 = \eta_2 = 0.5$

When the number of mixing components L is given a priori (where $L \neq \infty$), Equation (4.3) is particularly called *finite mixture model*, which is widely applied in various fields. In the case of mixture models for document vectors, it is assumed that each cluster C_k contributes to generation of vector representation $\mathbf{d}_i$ in proportion to size of parameter η_k.

For applying actually the mixture model, parameters contained in the model have to be estimated from observed data (i.e., document collections). A set of parameters to be estimated can be represented as a vector, which is denoted by $\mathbf{\Psi}$ in this section. For Poisson mixture model (PMM) based on Equation (4.2), it becomes

$$\mathbf{\Psi} = [\eta_1, \ldots, \eta_L, \lambda_{1|1} \ldots, \lambda_{M|L}]^T, \tag{4.5}$$

which includes $L + (L \times M) = (L + 1) \times M$ parameters since Poisson parameters $\lambda_{j|k}$ and mixing parameters η_k have to be estimated [1].

Estimation by EM algorithm

For estimating parameters in mixture models, it is feasible to employ an Expectation-Maximization algorithm (*EM algorithm*), which is "a broadly applicable approach to the iterative computation of maximum likelihood (ML) estimates, useful in a variety of incomplete-data problems, where algorithms such as the Newton-Raphson method may turn out to be more complicated" (p.1 in [208]). In general, the *likelihood function* can be defined as

$$\mathcal{L}(\mathbf{\Psi}) = P(\mathbf{y}; \mathbf{\Psi}), \tag{4.6}$$

[1] *Gaussian mixture model* (GMM) [111, 193] is based on a probabilistic density function

$$f(\mathbf{d}_i|C_k) = \frac{1}{(2\pi)^{M/2}|\mathbf{\Sigma}_k|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{d}_i - \boldsymbol{\mu}_k]^T \mathbf{\Sigma}_k^{-1} [\mathbf{d}_i - \boldsymbol{\mu}_k]\right),$$

where $\boldsymbol{\mu}_k$ denotes mean vector of k-th cluster and $\mathbf{\Sigma}_k$ indicates a covariance matrix ($\mathbf{\Sigma}_k^{-1}$ represents an inverse matrix, and $|\mathbf{\Sigma}_k|$ is a determinant). Although the GMM is widely used for various applications, it seems that there has been not so many cases utilizing directly it for DC. However, it should be noted that the k-means algorithm has a close relationship with the GMM-based clustering when every $\mathbf{\Sigma}_k$ is assumed to be a constant diagonal matrix, in which documents are grouped based on square Euclidean distance $[\mathbf{d}_i - \boldsymbol{\mu}_k]^T[\mathbf{d}_i - \boldsymbol{\mu}_k] = \|\mathbf{d}_i - \boldsymbol{\mu}_k\|^2$ [21]. In this case, $\mathbf{\Psi} = [\eta_1, \ldots, \eta_L, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_L,]^T$.

where $\mathbf{y}$ means observed data [2], namely, $\mathbf{y} = [\mathbf{d}_1^T, \ldots, \mathbf{d}_N^T]^T$ for DC problems. The *ML estimate* (MLE) of $\mathbf{\Psi}$ is obtained as a solution of equation $\partial \mathcal{L}(\mathbf{\Psi})/\partial \mathbf{\Psi} = \mathbf{0}$ [3], or equivalently,

$$\partial \log \mathcal{L}(\mathbf{\Psi})/\partial \mathbf{\Psi} = \mathbf{0}. \tag{4.7}$$

According to statistical theory, the MLE has some desirable properties, and can be considered as a 'standard' estimate when parameters in a probabilistic model have to be estimated from sample data.

However, $\mathcal{L}(\mathbf{\Psi})$ of mixture models is often too complicated to calculate the derivative in Equation (4.7). For instance, by assuming that documents are generated independently, $\mathcal{L}(\mathbf{\Psi})$ in DC situation can be written as

$$\mathcal{L}(\mathbf{\Psi}) = P(\mathbf{y}; \mathbf{\Psi}) = \prod_{i=1}^{N} P(\mathbf{d}_i) = \prod_{i=1}^{N} \sum_{k=1}^{L} \eta_k P(\mathbf{d}_i | C_k), \tag{4.8}$$

according to Equation (4.3). It is hard to differentiate analytically the most right-hand side of the equation when $P(\mathbf{d}_i|C_k)$ is a distribution describing term frequency like Figure 4.1. In such cases, the EM algorithm computes numerically the MLE of $\mathbf{\Psi}$ by assuming that $\mathbf{y}$ is *incomplete data* including partially missing values or unobservable variables. For mixture model problems, information on one-to-one relationship between a document (i.e., an element) and its cluster (i.e., a mixture component to which the element belongs) is assumed to be missing [208]. The relationship can be represented as unobservable variables (or missing data)

$$z_{ik} = \begin{cases} 1 & \text{if } i\text{-th document belongs to } k\text{-th cluster} \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

$(i = 1, \ldots, N;\ k = 1, \ldots, L)$, which are also denoted by $\mathbf{z} = [z_{11}, \ldots, z_{NL}]^T$.

The 'complete data' into which both the observed data and missing data are included can be written as $\mathbf{x} = [\mathbf{y}^T, \mathbf{z}^T]^T$ and the likelihood function of $\mathbf{x}$ is represented by $\mathcal{L}_c(\mathbf{\Psi}) = P(\mathbf{x}; \mathbf{\Psi})$. The unobservable variables are usually selected so that resulting $\mathcal{L}_c(\mathbf{\Psi})$ (or $\log \mathcal{L}_c(\mathbf{\Psi})$) becomes to be differentiated easily. This allows to obtain MLEs by maximizing $\mathcal{L}_c(\mathbf{\Psi})$, which is called M-step ('M' means the maximization). However, in order to calculate the derivative, it is necessary to allocate actual values to the unobservable variables beforehand. Roughly speaking, an expectation of the unobservable variable based on the observed data is used as its value (see below for precise explanations) [4]. The step of computing the expectation is called E-step ('E' indicates the expectation). The EM algorithm repeats iteratively the E- and M-steps until the unknown parameters in $\mathbf{\Psi}$ converge (note that each E-step is affected by a result from the previous M-step as shown in Equation (4.11)). If $\mathbf{\Psi}$ in s-th iterative step is denoted by $\mathbf{\Psi}^{(s)}$, then the EM algorithm allows $\mathbf{\Psi}^{(s)}$ for approaching gradually to the MLE of $\mathbf{\Psi}$ as s becomes larger.

Actually, in the case of mixture models in Equation (4.3), the log likelihood function for the complete data becomes

$$\log \mathcal{L}_c(\mathbf{\Psi}) = \sum_{k=1}^{L} \sum_{i=1}^{N} z_{ik} \log \eta_k + \sum_{k=1}^{L} \sum_{i=1}^{N} z_{ik} \log P(\mathbf{d}_i | C_k), \tag{4.10}$$

from Equations (4.8) and (4.9) because terms with $z_{ik} = 0$ vanish [5]. More precisely, in s-th E-step of the EM algorithm, it is necessary to compute the expectation

$$\mathcal{Q}(\mathbf{\Psi}; \mathbf{\Psi}^{(s)}) \equiv E_{\mathbf{\Psi}^{(s)}}\{\log \mathcal{L}_c(\mathbf{\Psi}) | \mathbf{y}\}, \tag{4.11}$$

[2] $P(\mathbf{y}; \mathbf{\Psi})$ is basically a probabilistic distribution of $\mathbf{y}$ based on parameters in $\mathbf{\Psi}$, which means that $P(\mathbf{y}; \mathbf{\Psi})$ is a function of $\mathbf{y}$ for fixed parameters. However, when $P(\mathbf{y}; \mathbf{\Psi})$ is interpreted as a likelihood function, it is considered as a function of $\mathbf{\Psi}$ given observed data $\mathbf{y}$, inversely.

[3] In this equation, $\mathbf{0}$ denotes a K-dimensional null vector $[0, \ldots, 0]^T$ where K indicates the number of parameters in $\mathbf{\Psi}$.

[4] For a probabilistic distribution $P(x)$ of a discrete random variable x, the expectation is defined as $E(x) = \sum_x xP(x)$ where $\sum_x$ denotes summation over all possible values of x. When x is a continuous variable, $E(x) = \int xf(x)dx$ where $f(x)$ is a probability density function.

[5] Since the logarithm is a monotonically increasing function, maximization of $\log f(x)$ is equal to that of $f(x)$.

where $E_{\mathbf{\Psi}^{(s)}}\{\mathbf{x}\}$ means an operation of expectation with respect to $\mathbf{x}$ by using parameters in $\mathbf{\Psi}^{(s)}$. Since $\mathcal{Q}(\mathbf{\Psi};\mathbf{\Psi}^{(s)})$ is a conditional expectation given $\mathbf{y}$, it follows that

$$E_{\mathbf{\Psi}_{(s)}}\{\log \mathcal{L}_c(\mathbf{\Psi})|\mathbf{y}\} = \sum_{\mathbf{Z}} \log \mathcal{L}_c(\mathbf{\Psi}^{(s)}) P_{\mathbf{\Psi}^{(s)}}(\mathbf{Z}|\mathbf{y}), \tag{4.12}$$

where $\mathbf{Z}$ is a vector of random variables corresponding to missing data $\mathbf{z}$, $P_{\mathbf{\Psi}^{(s)}}(\cdot)$ indicates the probability computed based on parameter $\mathbf{\Psi}^{(s)}$ (i.e., $P_{\mathbf{\Psi}^{(s)}}(\cdot) \equiv P(\cdot;\mathbf{\Psi}^{(s)})$), and $\sum_{\mathbf{Z}}$ denotes summation over all patterns of values to be taken as elements of vector $\mathbf{Z}$. In mixture models, $\mathbf{Z} = [Z_{11},\ldots,Z_{NL}]^T$ where Z_{ik} is a binary variable (i.e., $Z_{ik}=0$ or $Z_{ik}=1$), and also, $\sum_{\mathbf{Z}} = \sum_{Z_{11}}\sum_{Z_{12}}\cdots\sum_{Z_{NL}}$.

In s-th M-step, $\mathbf{\Psi}^{(s+1)}$ has to be determined so as to maximize $\mathcal{Q}(\mathbf{\Psi};\mathbf{\Psi}^{(s)})$, namely,

$$\mathcal{Q}(\mathbf{\Psi}^{(s+1)};\mathbf{\Psi}^{(s)}) \geq \mathcal{Q}(\mathbf{\Psi};\mathbf{\Psi}^{(s)}), \tag{4.13}$$

over all $\mathbf{\Psi} \in \mathbf{\Omega}$ where $\mathbf{\Omega}$ is a parameter space of $\mathbf{\Psi}$. If $\mathcal{Q}(\mathbf{\Psi}^{(s+1)};\mathbf{\Psi}^{(s)})$ satisfying the condition in Equation (4.13) is found, then the log likelihood $\mathcal{L}_c(\mathbf{\Psi}^{(s+1)})$ increases (or does not change) and its value is expected to approach somewhat to the maximum. Thus by repeating E-step in Equation (4.11) and M-step in Equation (4.13), $\mathbf{\Psi}^{(s)}$ would converge to the MLE of $\mathbf{\Psi}$ [6].

For computing $\mathcal{Q}(\mathbf{\Psi};\mathbf{\Psi}^{(s)})$ of mixture models, it is enough to calculate $P_{\mathbf{\Psi}^{(s)}}(Z_{ik}=1|\mathbf{y})$ and to replace z_{ik} in Equation (4.10) by it because Equation (4.10) is a linear function and Z_{ik} is a binary variable [7]. Namely if $z_{ik}^{(s)}$ denotes a value of z_{ik} in s-th step, then

$$z_{ik}^{(s)} = P_{\mathbf{\Psi}^{(s)}}(Z_{ik}=1|\mathbf{y}). \tag{4.14}$$

For showing it, each term corresponding to a particular pair of $\mathbf{d}_i$ and C_k in Equation (4.10) is represented by

$$g(z_{ik}) = z_{ik}\log\eta_k + z_{ik}\log P(\mathbf{d}_i|C_k). \tag{4.15}$$

Thus $\mathcal{Q}(\mathbf{\Psi};\mathbf{\Psi}^{(s)})$ based on Equation (4.10) can be written as

$$E_{\mathbf{\Psi}_{(s)}}\{\log \mathcal{L}_c(\mathbf{\Psi})|\mathbf{y}\} = \sum_{k=1}^{L}\sum_{i=1}^{N} E_{\mathbf{\Psi}_{(s)}}\{g(Z_{ik})|\mathbf{y}\}, \tag{4.16}$$

and each $E_{\mathbf{\Psi}_{(s)}}\{g(Z_{ik})|\mathbf{y}\}$ can be computed as

$$\begin{aligned}
E_{\mathbf{\Psi}_{(s)}}\{g(Z_{ik})|\mathbf{y}\} &= \sum_{\mathbf{Z}} g(Z_{ik}) P_{\mathbf{\Psi}^{(s)}}(\mathbf{Z}|\mathbf{y}) \\
&= \sum_{Z_{11}}\cdots\sum_{Z_{NL}} g(Z_{ik}) P_{\mathbf{\Psi}^{(s)}}(\mathbf{Z}|\mathbf{y}) \\
&= \sum_{Z_{ik}}\left[g(Z_{ik}) \sum_{\mathbf{Z}^{\neg ik}} P_{\mathbf{\Psi}^{(s)}}(\mathbf{Z}|\mathbf{y})\right] \\
&= \sum_{Z_{ik}=0}^{1} g(Z_{ik}) P_{\mathbf{\Psi}^{(s)}}(Z_{ik}|\mathbf{y}) \\
&= [\log\eta_k + \log P(\mathbf{d}_i|C_k)] P_{\mathbf{\Psi}^{(s)}}(Z_{ik}=1|\mathbf{y}),
\end{aligned} \tag{4.17}$$

where $\mathbf{Z}^{\neg ik}$ means a sequence of $Z_{11}\ldots Z_{NL}$ removing Z_{ik} [8]. Note that if $Z_{ik}=0$, then $g(Z_{ik})=0$ by definition in Equation (4.15). By substituting it into Equation (4.16), it turns out that Equation

[6] As to mathematical explanation why $\mathbf{\Psi}^{(s)}$ converges to the MLE, see p.78-79 in [208].

[7] For a linear function $f(x)=a+bx$, $E[f(x)] = E(a+bx) = E(a)+E(bx) = a+bE(x)$ by the definition of expectation. Also, if z is a binary variable, then $E(z) = E[zP(z)] = P(z)$ because a term with $z=0$ vanishes.

[8] For n-dimensional vector of probabilistic variables (denoted by $\mathbf{x}=[x_1,\ldots,x_n]^T$),

$$\sum_{x_2}\cdots\sum_{x_n} P(\mathbf{x}) = \sum_{\mathbf{x}^{\neg 1}} P(\mathbf{x}) = P(x_1),$$

which is called marginal distribution of $P(\mathbf{x})$.

(4.14) is valid. In particular, since $P_{\mathbf{\Psi}}(Z_{ik}=1|\mathbf{y}) = P_{\mathbf{\Psi}}(\mathbf{y}|Z_{ik}=1)P_{\mathbf{\Psi}}(Z_{ik}=1)/P_{\mathbf{\Psi}}(\mathbf{y})$ by Bayes' theorem, $z_{ik}^{(s)}$ can be updated by

$$z_{ik}^{(s)} = \frac{\eta_k^{(s)} P_{\mathbf{\Psi}^{(s)}}(\mathbf{d}_i|C_k)}{P_{\mathbf{\Psi}^{(s)}}(\mathbf{d}_i)} = \frac{\eta_k^{(s)} P_{\mathbf{\Psi}^{(s)}}(\mathbf{d}_i|C_k)}{\sum_{k'=1}^{L} \eta_{k'}^{(s)} P_{\mathbf{\Psi}^{(s)}}(\mathbf{d}_i|C_{k'})}, \tag{4.18}$$

according to Equation (4.3) [9] and $\eta_k^{(s)} = P_{\mathbf{\Psi}}(Z_{ik}=1)$ for all documents [10].

After computing the conditional expectation $\mathcal{Q}(\mathbf{\Psi}; \mathbf{\Psi}^{(s)})$ based on the updating formula of Equation (4.18) in E-step, $\mathcal{Q}(\mathbf{\Psi}^{(s+1)}; \mathbf{\Psi}^{(s)})$ satisfying the condition in Equation (4.13) has to be found in M-step. To do this for the mixture model, it is enough to differentiate the log likelihood function of the complete data in Equation (4.10) under an assumption that the missing values in $\mathbf{z}$ are completely known.

As to mixing parameter η_k, by introducing the Lagrange multiplier for the condition in Equation (4.4), an equation to be differentiated becomes

$$\mathcal{H} = \log \mathcal{L}_c(\mathbf{\Psi}) - \lambda \left(\sum_{k=1}^{L} \eta_k - 1 \right), \tag{4.19}$$

where λ denotes the Lagrange multiplier (not the Poisson parameter here), and it is necessary to solve equation $\partial\mathcal{H}/\partial\eta_k = 0$ $(k = 1, \ldots, L)$. Since $\partial\mathcal{H}/\partial\eta_k = \sum_i z_{ik}/\eta_k - \lambda$, the equation $\partial\mathcal{H}/\partial\eta_k = 0$ leads to $\eta_k = \lambda^{-1}\sum_i z_{ik}$. By definition in Equation (4.9), $\sum_k \sum_i z_{ik} = N$, and therefore, $\lambda = N$ for keeping the condition on mixing parameters in Equation (4.4). Thus the updating formula in M-step can be obtained as

$$\eta_k^{(s+1)} = \frac{1}{N} \sum_{i=1}^{N} z_{ik}^{(s)}, \quad k = 1, \ldots, L. \tag{4.20}$$

In addition to η_k, parameters for specifying probabilistic distribution $P(\mathbf{d}_i|C_k)$ have to be similarly estimated by using Equation (4.10) for executing M-step of the EM algorithm (see below). After the parameters converge in iterations of E- and M-steps, estimated values of z_{ik} in Equation (4.18) can be directly used for clustering. Specifically, it is feasible to partition the target document set such that

$$C_k = \left\{ d_i \,\middle|\, \arg\max_{k'} z_{ik'} = k \right\}, \qquad k = 1, \ldots, L. \tag{4.21}$$

Needless to say, results of DC based on the estimation of missing values in $\mathbf{z}$ may vary with the probabilistic distribution $P(\mathbf{d}_i|C_k)$ to be used in the EM algorithm. In the following sections, Poisson, multinomial and von Mises-Fisher distributions will be actually employed as $P(\mathbf{d}_i|C_k)$.

EM algorithm for Poisson mixture model

When $P(\mathbf{d}_i|C_k)$ is described as a Poisson distribution in Equation (4.1), Equation (4.18) in E-step becomes

$$z_{ik}^{(s)} = \frac{\eta_k^{(s)} \prod_j \lambda_{j|k}^{(s)\mathrm{f}_{ij}} \exp(-\lambda_{j|k}^{(s)})}{\sum_{k'=1}^{L} \eta_{k'}^{(s)} \prod_j \lambda_{j|k'}^{(s)\mathrm{f}_{ij}} \exp(-\lambda_{j|k'}^{(s)})}, \tag{4.22}$$

since $\prod_j \mathrm{f}_{ij}!$ in Equation (4.1) is canceled out. For obtaining an estimation of $\lambda_{j|k}^{(s+1)}$ in M-step, $\log \mathcal{L}_c(\mathbf{\Psi})$ in Equation (4.10) has to be differentiated. So,

$$\frac{\partial \log \mathcal{L}_c(\mathbf{\Psi})}{\partial \lambda_{j|k}} = \sum_{i=1}^{N} \frac{z_{ik}}{P(\mathbf{d}_i|C_k)} \frac{\partial P(\mathbf{d}_i|C_k)}{\partial \lambda_{j|k}} \tag{4.23}$$

[9] Note that $P_{\mathbf{\Psi}}(Z_{ik}=1|\mathbf{y}) = P_{\mathbf{\Psi}}(Z_{ik}=1|\mathbf{d}_i)$ for particular d_i if it is assumed that documents in D are generated independently.

[10] Probability $P_{\mathbf{\Psi}}(Z_{ik}=1)$ is not dependent on a particular document.

Table 4.1: Estimation of $\lambda_{j|k}$ in PMM on sample DB

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$k=1$	2.000	3.333	0.000	0.000	0.000	0.000
$k=2$	0.000	0.987	2.961	0.760	1.779	0.247
$k=3$	0.000	0.000	0.000	2.009	2.982	1.358

Note: $L=3$ and $\log \mathcal{L}(\mathbf{\Psi}) = -1254.59$.

where

$$\frac{\partial P(\mathbf{d}_i|C_k)}{\partial \lambda_{j|k}} = \frac{P(\mathbf{d}_i|C_k)}{P(\mathrm{f}_{ij}|C_k)} \frac{\partial P(\mathrm{f}_{ij}|C_k)}{\partial \lambda_{j|k}}, \tag{4.24}$$

according to the term independence model in Equation (4.2) [11]. Since

$$\begin{aligned} \frac{\partial P(\mathrm{f}_{ij}|C_k)}{\partial \lambda_{j|k}} &= \frac{\mathrm{f}_{ij} \lambda_{j|k}^{\mathrm{f}_{ij}-1} e^{-\lambda_{j|k}}}{\mathrm{f}_{ij}!} - \frac{\lambda_{j|k}^{\mathrm{f}_{ij}} e^{-\lambda_{j|k}}}{\mathrm{f}_{ij}!} \\ &= P(\mathrm{f}_{ij}|C_k) \times \left(\frac{\mathrm{f}_{ij}}{\lambda_{j|k}} - 1 \right), \end{aligned} \tag{4.25}$$

Equation(4.23) becomes

$$\frac{\partial \log \mathcal{L}_c(\mathbf{\Psi})}{\partial \lambda_{j|k}} = \sum_{i=1}^{N} z_{ik} \left(\frac{\mathrm{f}_{ij}}{\lambda_{j|k}} - 1 \right), \tag{4.26}$$

by substituting Equations (4.24) and (4.25) into it [12]. Thus the updating formula for M-step,

$$\lambda_{j|k}^{(s+1)} = \frac{1}{\sum_{i=1}^{N} z_{ik}^{(s)}} \sum_{i=1}^{N} z_{ik}^{(s)} \mathrm{f}_{ij}, \tag{4.27}$$

can be obtained from the equation that $\partial \log \mathcal{L}_c(\mathbf{\Psi}) / \partial \lambda_{j|k} = 0$ [13].

A summarization of the EM algorithm for PMM is shown as Figure 4.2. Also, Tables 4.1 and 4.2 indicate a result of estimating actually $\lambda_{j|k}$ and z_{ik}, respectively, when the EM algorithm for PMM was executed for the sample DB of Table 3.1 in the case that $L=3$ [14]. In the execution, η_k was estimated such that $\hat{\eta}_1 = 0.3$, $\hat{\eta}_2 = 0.41$ and $\hat{\eta}_3 = 0.29$. Consequently, according to the estimation of z_{ik}, the set of clusters generated by PMM was {{d_1, d_2, d_3},{d_4, d_5, d_6, d_7},{d_8, d_9, d_{10}}}.

In this execution on the sample DB, log likelihood scores $\log \mathcal{L}(\mathbf{\Psi})$ computed by using estimated parameters converged after around 20 iterations. More specifically, Figure 4.3 shows change of the score during the iterations, which indicates that the score rapidly increased in the beginning several iterations. Note that this is only an example, and that the number of iterations up to enough convergence is dependent on initial values assigned to parameters.

[11] When $y = \sum_i f(x_i)$, $dy/dx_k = df(x_k)/dx_k$ (which is called 'termwise differentiation'). If $f(x_k)$ can be formulated as $f(g(x_k))$, then $df(x_k)/dx_k = (df/dg) \times (dg/dx_k)$ by chain rule. In the case that $f(x_k) = \log g(x_k)$, the differential becomes $df(x_k)/dx_k = g(x_k)^{-1} \times dg(x_k)/dx_k$, which is often needed for differentiating log likelihood functions (note that differential of $\log x$ is x^{-1}). Also, when $y = \prod_i x_i$, $dy/dx_k = y/x_k$.

[12] When $y = f(x)g(x)$, $dy/dx = (df(x)/dx)g(x) + f(x)(dg(x)/dx)$. Also, differential of e^x becomes again e^x.

[13] According to statistical theory, the MLE of Poisson parameter is sample mean. Naturally, Equation (4.27) coincides with an average of frequencies of term t_j in documents of cluster C_k when z_{ik} is completely observed because $\sum_i z_{ik}$ represents the number of documents included in cluster C_k and similarly $\sum_i z_{ik} \mathrm{f}_{ij}$ means the total occurrence frequency of term t_j in C_k.

[14] In order to prevent $\lambda_{j|k}$ from becoming zero, the updating formula in M-step was slightly changed to

$$\lambda_{j|k}^{(s+1)} = \frac{1}{\sum_{i=1}^{N} z_{ik}^{(s)}} \left(\epsilon + \sum_{i=1}^{N} z_{ik}^{(s)} \mathrm{f}_{ij} \right),$$

where ϵ is a very small number (e.g., $\epsilon = 1.0 \times 10^{-8}$). Also, Math.random() method of Java language was used for initialization of parameters in $\mathbf{\Psi}$. Particularly, in the case of η_k, after a random number was allocated to each of $\eta_1, \ldots, \eta_L$, these values were normalized as $\eta_k / \sum_{k'} \eta_{k'}$ in this experiment. Similar methods for initialization of parameters were used in other experiments discussed in this chapter.

‡ Document clustering by PMM

Set: The number of clusters L.

1) Initialize parameters η_k and $\lambda_{j|k}$ $(k = 1, \ldots, L; j = 1, \ldots, M)$ by generating random numbers.

2) [E-step] Compute z_{ik} by Equation (4.22) $(k = 1, \ldots, L; i = 1, \ldots, N)$

3) [M-step] Update η_k and $\lambda_{j|k}$ by Equations (4.20) and (4.27) $(k = 1, \ldots, L;\ j = 1, \ldots, M)$.

4) If parameters converge, then allocate each document to a cluster based on z_{ik} using Equation (4.21) and terminate the procedure. Otherwise, return to 2).

Out: Clusters $C_1, \ldots, C_L$ and estimated values of parameters.

Figure 4.2: Document clustering by PMM

Table 4.2: Estimation of z_{ik} in PMM model on sample DB

	$i =$									
	1	2	3	4	5	6	7	8	9	10
$k = 1$	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$k = 2$	0.00	0.00	0.00	1.00	1.00	1.00	1.00	0.03	0.03	0.00
$k = 3$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.97	0.97	1.00

Note: $L = 3$ and $\log \mathcal{L}(\mathbf{\Psi}) = -1254.59$.

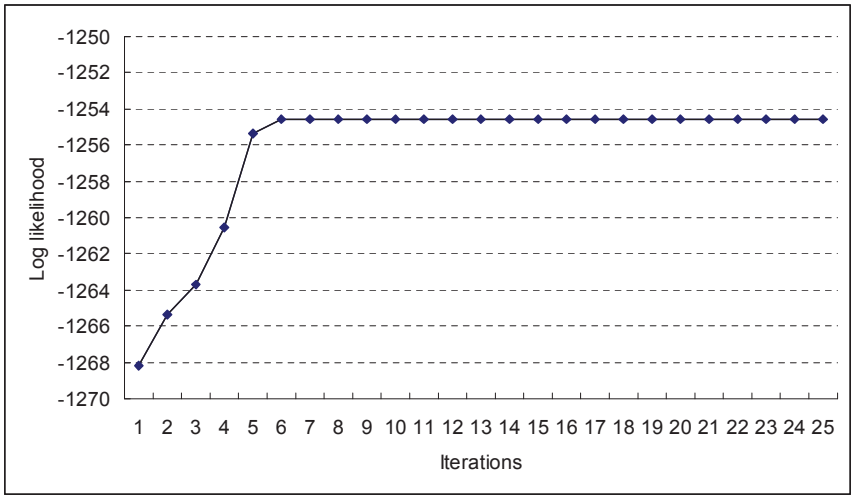


Figure 4.3: Converge of the log likelihood score in an experiment

Unfortunately, particular initial values may lead an iterative computation to a 'local maximum' of the log likelihood in a similar way of obtaining local minimums of the objective function for k-means algorithms (see Figure 3.25). Figure 4.4 shows an example of local maximums provided by x_1 and x_3, respectively, other than 'global maximum' by x_2. As the figure indicates, when the objective function maximized by an EM algorithm is not unimodal, its estimation may reach to a local maximum. Inevitably, parameter estimations on local maximums differ from that on the global maximum, which may cause a problem in applying mixture models.

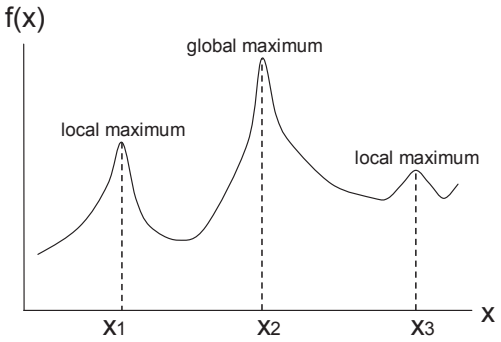


Figure 4.4: Global and local maximums

For understanding the local maximum problem in DC, an experimental result of repeating the estimation 1000 times with changing randomly initial parameters is shown in Figure 4.5. In the simple experiment, 38 different maximums appeared (other maximums may be obtained if the estimation will be repeated more). This suggests that the likelihood function of PMM for DC has a complicated and multi-modal shape. Among them, the largest score was -1254.59, which corresponds to the most left-hand bar in the graph since the maximums are arranged in descending order. Note that the result of estimations in Tables 4.1 and 4.2 was obtained when $\log \mathcal{L}(\mathbf{\Psi}) = -1254.59$.

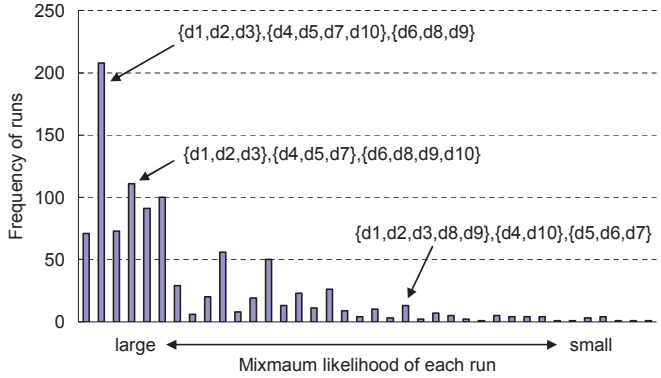


Figure 4.5: Result of repeated estimations by EM algorithm of PMM (1000 runs)

In Figure 4.5, the two most appearing maximums were
$\log \mathcal{L}(\mathbf{\Psi}) = -1254.83$: 208 times,
$\log \mathcal{L}(\mathbf{\Psi}) = -1255.09$: 111 times,
which provided 'valid' clustering results that were obtained by the k-means algorithm and so on (the sets of clusters are displayed in Figure 4.5). In contrast, small maximums generated 'invalid' clusters. For example, when $\log \mathcal{L}(\mathbf{\Psi}) = -1264.72$, the set $\{\{d_1, d_2, d_3, d_8, d_9\}, \{d_4, d_{10}\}, \{d_5, d_6, d_7\}\}$ was provided (see Figure 4.5). Although a 'valid' cluster set was not always generated for the sample DB even if the k-means algorithm was applied as described in Section 3.3.1, the probability of obtaining the 'valid' cluster set appears to be lower in the case of PMM in comparison with the Hartigan-Wong algorithm (Table 3.11).

Multinomial mixture model

Alternative way for computing $P(\mathbf{d}_i|C_k)$ in DC (where $\mathbf{d}_i = [\mathrm{f}_{i1}, \ldots, \mathrm{f}_{iM}]^T$ again) would be to use *multinomial distribution* [223, 233] such that

$$P(\mathbf{d}_i|C_k) = A_i \times \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}}, \tag{4.28}$$

where $p_{j|k}$ is a parameter constrained such that $\sum_j p_{j|k} = 1$ ($j = 1, \ldots, M$; $k = 1, \ldots, L$), and A_i denotes a multinomial coefficient [15] for document d_i. The parameter $p_{j|k}$ can be interpreted as a probability that term t_j is used in cluster C_k. In other words, document representation $\mathbf{d}_i$ is assumed to be generated in cluster C_k by selecting randomly term t_j according to probability $p_{j|k}$ in Equation (4.28). Thus the vector of parameters becomes

$$\mathbf{\Psi} = [\eta_1, \ldots, \eta_L, p_{1|1}, \ldots, p_{M|L}]^T, \tag{4.29}$$

in the case of *multinomial mixture model* (MMM). Note that the number of parameters in the MMM is equal to that in the PMM (i.e, $(L+1) \times M$). Multinomial distribution is often used in text categorization, and in this sense, it may be a natural choice for DC [16].

Evidently, Equation (4.20) is again used for updating mixing parameters η_k in M-step of an EM algorithm for the MMM. Meanwhile, since Equation (4.28) is substituted into Equation (4.18) for updating unobservable variable z_{ik} in E-step, it becomes

$$z_{ik}^{(s)} = \frac{\eta_k^{(s)} \prod_j p_{j|k}^{(s)\mathrm{f}_{ij}}}{\sum_{k'=1}^{L} \eta_{k'}^{(s)} \prod_j p_{j|k'}^{(s)\mathrm{f}_{ij}}} \tag{4.30}$$

(note that multinomial coefficient A_i in Equation (4.28) is canceled out). Finally, as updating formula of $p_{j|k}$ in M-step,

$$p_{j|k}^{(s+1)} = \frac{\sum_{i=1}^{N} z_{ik}^{(s)} \mathrm{f}_{ij}}{\sum_{j'=1}^{M} \sum_{i=1}^{N} z_{ik}^{(s)} \mathrm{f}_{ij'}}, \tag{4.31}$$

can be derived by differentiating again an equation,

$$\mathcal{H} = \mathcal{L}_c(\mathbf{\Psi}) - \lambda \left(\sum_{j=1}^{M} p_{j|k} - 1 \right), \tag{4.32}$$

with respect to each $p_{j|k}$ like Equation (4.19). Actually, because

$$\frac{\partial P(\mathbf{d}_i|C_k)}{\partial p_{j|k}} = \frac{P(\mathbf{d}_i|C_k)}{p_{j|k}^{\mathrm{f}_{ij}}} \times \mathrm{f}_{ij} p_{j|k}^{\mathrm{f}_{ij}-1} = \frac{\mathrm{f}_{ij} P(\mathbf{d}_i|C_k)}{p_{j|k}}, \tag{4.33}$$

in the case of Equation (4.28), an equation,

$$\frac{\partial \mathcal{H}}{\partial p_{j|k}} = \sum_{i=1}^{N} \frac{z_{ik}}{P(\mathbf{d}_i|C_k)} \frac{\partial P(\mathbf{d}_i|C_k)}{\partial p_{j|k}} - \lambda = \sum_{i=1}^{N} \frac{z_{ik} \mathrm{f}_{ij}}{p_{j|k}} - \lambda, \tag{4.34}$$

can be obtained. By solving $\partial \mathcal{H} / \partial p_{j|k} = 0$, it follows that $p_{j|k} = \lambda^{-1} \sum_i z_{ik} \mathrm{f}_{ij}$. Since λ is a constant for keeping the condition that $\sum_j p_{j|k} = 1$ (i.e., λ is a normalizing factor), Equation (4.31) is finally obtained. Of course, Equation (4.31) is a standard MLE of the parameter in

[15] More precisely,

$$A_i = \frac{l_i!}{\mathrm{f}_{i1}! \mathrm{f}_{i2}! \ldots \mathrm{f}_{iM}!}$$

where $l_i = \sum_j \mathrm{f}_{ij}$ (i.e., document length as before).

[16] Naive Bayes classifier for categorizing documents into a given classification scheme is often constructed based on the multinomial model.

Table 4.3: Values of $p_{j|k}$ in the sample database

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$k=1$	0.375	0.625	0.000	0.000	0.000	0.000
$k=2$	0.000	0.146	0.439	0.113	0.265	0.037
$k=3$	0.000	0.000	0.000	0.316	0.470	0.214

Note: $\log L(\mathbf{\Psi}) = -46.82$

multinomial distribution under an assumption that z_{ik} is completely known (i.e., the numerator means occurrence frequency of t_j within cluster C_k and the denominator is its total over all terms).

Consequently, in the EM algorithm for MMM, Equations (4.22) and (4.27) in the EM algorithm of Figure 4.2 are replaced by Equations (4.30) and (4.31), respectively, while the other parts are not changed except that the notation $\lambda_{j|k}$ is rewritten as $p_{j|k}$ [17]. As mentioned before, in both the PMM and MMM, the number of unknown parameters amounts to $L + L \times M = (L+1) \times M$ (see Equations (4.5) and (4.29)). Also, for calculating all values of z_{ik} and $\lambda_{j|k}$ (or $p_{j|k}$) in both the models, $L \times M \times N$ computations in loops of indexes i, j and k are needed in each E- and M-step, respectively. Thus if r iterations are required for convergence of estimates, then computational complexity of the PMM and MMM becomes $O(rLMN)$, which means that their complexity is comparable with that of the k-means algorithm.

In terms of not only computational complexity but also cluster validity, the PMM and MMM would be similar. When the MMM is applied to the sample DB ($L = 3$), $\log \mathcal{L}(\mathbf{\Psi}) = -46.82$ is considered to be larger than any other local maximums. Table 4.3 shows estimates of $p_{j|k}$ in the case of the largest maximum likelihood score. As to z_{ik}, the values were almost same with those obtained by the PMM shown in Table 4.2, which means that the PMM and MMM provided the same clustering result in the global maximum for the sample DB. Similarly, estimates of η_k were almost equal between the two models.

Also, like the situation represented by Figure 4.5 for the PMM, 33 distinct maximums appeared in an experiment executing repeatedly 1000 runs of the EM algorithm for MMM with different initial values. The second largest maximum ($\log \mathcal{L}(\mathbf{\Psi}) = -47.17$) and the fourth maximum ($\log \mathcal{L}(\mathbf{\Psi}) = -47.30$) provided sets of clusters, $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7, d_{10}\}, \{d_6, d_8, d_9\}\}$ and $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$, respectively. They are the most frequently appearing maximums in the experiment (i.e., 266 and 153, respectively [18]). While a result that the second and forth largest maximums with top frequencies generated 'valid' cluster sets was the same with that in the case of PMM, the probability of obtaining the 'valid' sets by the MMM seems to be slightly higher than by the PMM in this experiment. Section 4.1.2 will try to solve the problem of many local maximums in the MMM by introducing a Bayesian framework.

Mixture of von Mises-Fisher distributions

One of the motivations for exploring spherical k-means methods [81] in which inner product is used for measuring similarities is that computational efficiency of the inner product is higher than that of the Euclidean distance when document vectors are high-dimensional and sparse (see Section 3.3.1). In this sense, *von Mises-Fisher distribution* playing an important role in the field of directional statistics [19] can be considered as a candidate of $P(\mathbf{d}_i|C_k)$ in mixture models [20, 19] because its central part is an inner product of a random variable vector and a parameter vector.

When document vectors are normalized so that $||\tilde{\mathbf{d}}_i|| = 1$, they are placed on a circle if $M = 2$ (see Figure 3.2). In general, the unit vectors are distributed on the surface of an M-dimensional sphere (hypersphere). Such kind of data on a sphere can be analyzed by the theory of directional statistics, and the von Mises-Fisher (vMF) distribution would be a natural choice for treating unit

[17] Like the algorithm for PMM, it may be better to replace $\sum_i z_{ik} \mathrm{f}_{ij}$ in Equation (4.31) by $\epsilon + \sum_i z_{ik} \mathrm{f}_{ij}$ in order to prevent $p_{j|k}$ from becoming zero where ϵ is a very small number.

[18] See the most left-hand column of Table 4.9. In Bayesian model of multinomial mixture, the standard MMM discussed here corresponds to that with $\alpha = 1.0$ and $\beta = 1.0$ (see Section 4.1.2 for details).

[19] Examples of cases treated by directional statistics are vanishing angle of birds released at a site, orientation of turtles after laying eggs, and so on [203].

Table 4.4: Probabilistic densities of vMF distribution

	$(1)d_1$	$(2)d_2$	$(3)d_3$	$(4)d_5$	$(2)/(1)$	$(2)/(3)$
$\kappa_1 = 1$	0.06	0.08	0.07	0.05	1.25	1.05
$\kappa_1 = 5$	0.23	0.72	0.55	0.09	3.08	1.30
$\kappa_1 = 10$	0.29	2.74	1.62	0.05	9.50	1.70
$\kappa_1 = 20$	0.11	9.85	3.43	0.00	90.22	2.87
$\kappa_1 = 50$	0.00	31.99	2.28	0.00	77316.16	14.01
$\|\tilde{\mathbf{d}}_i - \boldsymbol{\mu}_1\|$	0.72	0.27	0.42	0.94		

Note: It is assumed that $C_1 = \{d_1, d_2, d_3\}$.

vectors $\tilde{\mathbf{d}}_i$ $(i = 1, \ldots, N)$.

Specifically, $P(\mathbf{d}_i|C_k)$ in the mixture model of vMF distributions is written as

$$P(\tilde{\mathbf{d}}_i|C_k) = c_M(\kappa_k) \exp[\kappa_k \boldsymbol{\mu}_k^T \tilde{\mathbf{d}}_i], \tag{4.35}$$

where

$$c_M(\kappa_k) = \frac{\kappa_k^{M/2-1}}{(2\pi)^{M/2} \mathcal{B}_{M/2-1}(\kappa_k)}, \tag{4.36}$$

κ_k is a 'concentration parameter' $(k = 1, \ldots, L)$, and $\mathcal{B}_{M/2-1}(\kappa_k)$ is the modified Bessel function of the first kind and order of $M/2 - 1$. Also, it is necessary that $\|\boldsymbol{\mu}_k\| = 1$ $(k = 1, \ldots, L)$. Note that Equation (4.35) shows probabilistic density function of the vMF distribution. Because $\tilde{\mathbf{d}}_i$ and $\boldsymbol{\mu}_k$ are normalized, $\boldsymbol{\mu}_k^T \tilde{\mathbf{d}}_i$ in Equation (4.35) can be interpreted as a cosine coefficient measuring similarity between them.

Suppose that $C_1 = \{d_1, d_2, d_3\}$ and $\boldsymbol{\mu}_1 = \sum_{i:d_i \in C_1} \tilde{\mathbf{d}}_i / \| \sum_{i:d_i \in C_1} \tilde{\mathbf{d}}_i \|$. Table 4.4 shows values of the probabilistic density in Equation (4.35) for d_1, d_2, d_3 and d_5. In the situation, d_2 is geometrically the closest to $\boldsymbol{\mu}_1$ and its value becomes larger as the concentration parameter κ_1 increases. Namely, when κ_1 is large, probability mass tends to concentrate on points near $\boldsymbol{\mu}_1$.

In the case of vMF distribution, the log likelihood of Equation (4.10) has to be maximized with respect to $\boldsymbol{\mu}_k$ and κ_k $(k = 1, \ldots, L)$ under a condition that $\boldsymbol{\mu}_k^T \boldsymbol{\mu}_k = 1$. Thus an equation to be maximized for estimating $\boldsymbol{\mu}_k$ and κ_k in M-step becomes

$$\mathcal{H} = \sum_{k=1}^{L} \left[\sum_{i=1}^{N} z_{ik} \log c_M(\kappa_k) + \sum_{i=1}^{N} z_{ik} \kappa_k \boldsymbol{\mu}_k^T \tilde{\mathbf{d}}_i + \lambda_k (1 - \boldsymbol{\mu}_k^T \boldsymbol{\mu}_k) \right], \tag{4.37}$$

where λ_k is the Lagrange multiplier for C_k $(k = 1, \ldots, L)$. Because

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{\mu}_k} = \sum_{i=1}^{N} z_{ik} \kappa_k \tilde{\mathbf{d}}_i - 2\lambda_k \boldsymbol{\mu}_k, \tag{4.38}$$

each cluster vector can be obtained as

$$\boldsymbol{\mu}_k = \frac{\kappa_k}{2\lambda_k} \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i, \tag{4.39}$$

by solving $\partial \mathcal{H} / \partial \boldsymbol{\mu}_k = 0$ [20]. Similarly, from a differentiation of $\mathcal{H}$ with respect to κ_k,

$$\frac{\partial \mathcal{H}}{\partial \kappa_k} = \sum_{i=1}^{N} \frac{c'_M(\kappa_k)}{c_M(\kappa_k)} z_{ik} + \sum_{i=1}^{N} z_{ik} \boldsymbol{\mu}_k^T \tilde{\mathbf{d}}_i, \tag{4.40}$$

where $c'_M(\kappa_k)$ denotes differential of $c_M(\kappa_k)$, and an equation $\partial \mathcal{H} / \partial \kappa_k = 0$ provides

$$\frac{c'_M(\kappa_k)}{c_M(\kappa_k)} \sum_{i=1}^{N} z_{ik} = -\boldsymbol{\mu}_k^T \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i. \tag{4.41}$$

[20] When $f(\mathbf{x}) = \mathbf{x}^T \mathbf{x}$, $\partial f(\mathbf{x}) / \partial \mathbf{x} = 2\mathbf{x}$.

From the condition that $\boldsymbol{\mu}_k^T \boldsymbol{\mu}_k = 1$ and Equation (4.39), it follows that

$$\left[\frac{\kappa_k}{2\lambda_k} \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i \right]^T \left[\frac{\kappa_k}{2\lambda_k} \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i \right] = 1, \tag{4.42}$$

and therefore,

$$\lambda_k = \frac{\kappa_k \| \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i \|}{2}, \tag{4.43}$$

and

$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i}{\| \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i \|}. \tag{4.44}$$

Also, by substituting Equation (4.44) into Equation (4.41), it becomes that

$$\frac{c'_M(\kappa_k)}{c_M(\kappa_k)} = -\frac{\| \sum_{i=1}^{N} z_{ik} \tilde{\mathbf{d}}_i \|}{\sum_{i=1}^{N} z_{ik}}. \tag{4.45}$$

By using Equation (4.45), the concentration parameter κ_k $(k = 1, \ldots, L)$ can be estimated such that

$$\hat{\kappa}_k = \frac{\bar{A}M - \bar{A}^3}{1 - \bar{A}^2}, \tag{4.46}$$

where $\bar{A} = -c'_M(\kappa_k)/c_M(\kappa_k)$ [19] [21]. The resulting procedure of an EM algorithm for estimating mixture model of vMF distributions for DC is shown in Figure 4.6.

‡ Document clustering by vMF mixture model

Set: The number of clusters L.

1) Initialize the parameters η_k, $\boldsymbol{\mu}_k$ and κ_k $(k = 1, \ldots, L)$ by generating random numbers.

2) [E-step] Compute z_{ik} by Equations (4.35) and (4.18) $(i = 1, \ldots, N;\ k = 1, \ldots, L)$.

3) [M-step] Update η_k, μ_k and κ_k by Equations (4.20), (4.44) and (4.46), respectively $(k = 1, \ldots, L)$.

4) If parameters converge, then allocate each document to a cluster based on z_{ik} using Equation (4.21) and terminate the procedure. Otherwise, return to 2).

Out: Clusters $C_1, \ldots, C_L$ and estimated values of parameters.

Figure 4.6: Document clustering by vMF mixture model

Table 4.5 indicates a result of a simple experiment in which clustering of the sample DB by the algorithm in Figure 4.6 was repeated 1000 times with different initial parameters. Actually, because it is possible that an estimated value of κ_k becomes infinite [18] in the process, '1000.0' was arbitrarily used as an upper bound of κ_k in this experiment. As the table indicates, although nine maximums appeared in the repeated runs, about 90% of them generated sets of clusters, $\{\{d_1, d_2, d_3\},\ \{d_4, d_5, d_7, d_{10}\},\ \{d_6, d_8, d_9\}\}$ or $\{\{d_1, d_2, d_3\},\ \{d_4, d_5, d_7\},\ \{d_6, d_8, d_9, d_{10}\}\}$, which are 'valid' sets of clusters. For example, the document set at maximum No.1 in Table 4.5 was obtained from estimated values of z_{ik} shown in Table 4.6. Due to the fact that probability of obtaining the 'valid' set is near 0.9, the experimental result shows that the vMF mixture model may be more useful than the simple PMM and MMM.

Also, in the case of maximum No.1, the concentration parameters were estimated such as $\hat{\kappa}_1 = 19.02$, $\hat{\kappa}_2 = 10.05$, and $\hat{\kappa}_3 = 66.39$, and as to mixing parameters, $\hat{\eta}_1 = 0.3$, $\hat{\eta}_2 = 0.4$ and $\hat{\eta}_3 = 0.3$. Also, final values of $\boldsymbol{\mu}_k$ are shown in Table 4.7.

[21] There are other methods for estimating the concentration parameter of the vMF distribution (see [18]).

Table 4.5: Result of estimation by EM algorithm of vMF mixture model

No.	# of runs	Clusters ($L = 3$)
1	637	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7, d_{10}\}\{d_6, d_8, d_9\}$
2	260	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$
3	39	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7\}\{d_6, d_8, d_9, d_{10}\}$
4	23	$\{d_1, d_2, d_3, d_{10}\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9\}$
5	23	$\{d_1, d_2, d_3, d_5, d_{10}\}\{d_4, d_7\}\{d_6, d_8, d_9\}$
6	15	$\{d_1, d_2, d_3\}\{d_4, d_6, d_8, d_9, d_{10}\}\{d_5, d_7\}$
7	1	$\{d_1, d_2, d_3, d_4, d_5, d_7\}\{d_6, d_9\}\{d_8, d_{10}\}$
8	1	$\{d_1, d_2, d_3, d_4, d_{10}\}\{d_5, d_7\}\{d_6, d_8, d_9\}$
9	1	$\{d_1, d_{10}\}\{d_2, d_3, d_4, d_5, d_7\}\{d_6, d_8, d_9\}$
Total	1000	

Note: In total, 1000 runs were repeated for the sample DB.

Table 4.6: Estimation of z_{ik} in vMF mixture model

	$i =$									
	1	2	3	4	5	6	7	8	9	10
$k = 1$	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$k = 2$	0.00	0.00	0.00	1.00	1.00	0.00	1.00	0.00	0.00	1.00
$k = 3$	0.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00

Note: In the case of maximum No.1 in Table 4.5.

Table 4.7: Estimation of $\boldsymbol{\mu}_k$ in vMF mixture model

	$j = 1$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$
$k = 1$	0.553	0.833	0.002	0.000	0.001	0.000
$k = 2$	0.001	0.278	0.785	0.388	0.234	0.317
$k = 3$	0.000	0.067	0.067	0.260	0.961	0.000

Note: In the case of maximum No.1 in Table 4.5.

Model-based deterministic annealing

For a joint probability of two discrete random variables x and y, it is feasible to consider an expectation $E[\log P(x|y)] = \sum_x \sum_y P(x,y) \log P(x|y) = \sum_x P(x) \sum_y P(y|x) \log P(x|y)$, which corresponds to the expected log likelihood of $P(x|y)$. By adding 'entropy constraint' to it, an objective function,

$$\log \mathcal{L}_1 = E[\log P(x|y)] - \mathrm{T} \times M_I(x,y), \tag{4.47}$$

can be obtained where $M_I(x,y)$ is the mutual information (MI) and T is a parameter that controls balance between maximization of $E[\log P(x|y)]$ and minimization of $M_I(x,y)$. Since a constraint $\sum_y P(y|x) = 1$ has to be incorporated, the target function becomes finally

$$\mathcal{H} = \log \mathcal{L}_1 + \sum_x \lambda_x \left(\sum_y P(y|x) - 1 \right), \tag{4.48}$$

where λ_x is the Lagrange multiplier. An equation $\partial \mathcal{H} / \partial P(y|x) = 0$ leads to

$$P(y|x) = \frac{P(y)P(x|y)^{1/\mathrm{T}}}{\sum_{y'} P(y')P(x|y')^{1/\mathrm{T}}}, \tag{4.49}$$

which can be interpreted as a *Gibbs distribution* [22]. In *simulated annealing* which is a method used extensively for optimization problems, T is interpreted as 'temperature'. Note that $P(y)$ can be estimated by $P(y) = \sum_x P(x)P(y|x)$.

In *model-based deterministic annealing* for DC [341, 342], x is replaced by document vector $\mathbf{d}_i$ and y is considered as cluster C_k. So $P(x|y)$ becomes $P(\mathbf{d}_i|C_k)$ which is a probabilistic model such as multinomial, vMF and so on, and its parameters can be estimated by repeating E-step and M-step of an EM algorithm again. Precisely, after Equation (4.49) is computed [23], parameters in the probabilistic model are calculated so as to maximize $\sum_x P(y|x) \log P(x|y)$ for each cluster (note that $P(x|y) = P(\mathbf{d}_i|C_k)$, which is a probabilistic model). If the parameters converge, then each document is assigned to a cluster whose $P(y|x)$ (i.e., $= P(C_k|\mathbf{d}_i)$) is the maximum.

During the iterated calculations, the temperature becomes lower gradually. For instance, T decreases such that $\mathrm{T}^{(s+1)} = a\mathrm{T}^{(s)}$ in s-th step where $0 < a < 1$. When the temperature is lower, $P(x|y)^{1/\mathrm{T}}$ of a dominant y (i.e., the cluster whose $P(\mathbf{d}_i|C_k)$ is the largest) becomes relatively larger than those of the others due to the exponential factor $1/\mathrm{T}$. Extremely, if $\mathrm{T} \to 0$, then it becomes

$$P(y|x) = \begin{cases} 1, & y = \arg\max_{y'} \log P(x|y') \\ 0, & \text{otherwise} \end{cases}, \tag{4.50}$$

which implies that a k-means algorithm is executed by assigning a document to a cluster with the highest score computed from the model $P(\mathbf{d}_i|C_k)$, not with the shortest Euclidean distance to its centroid, and by updating parameters in $P(\mathbf{d}_i|C_k)$ based on the assignment so that $P(\mathbf{d}_i|C_k)$ is maximized [24]. This can be called *model-based k-means algorithm*. In contrast, since the exponential factor vanishes with $\mathrm{T} = 1$, Equation (4.49) leads to a standard EM algorithm in this case. Therefore, the model-based deterministic annealing may be considered to provide a unified framework of clustering based on a probabilistic model [341].

[22] $M_I(x,y)$ can be rewritten as $\sum_x \sum_y P(y|x)P(x) \log P(y|x)/P(y)$. If putting that $f = P(y|x)P(x)$ and $g = \log P(y|x)/P(y)$, then $f' = P(x)$ and $g' = P(y|x)^{-1}$ where f' and g' denote differentiation with respect to $P(y|x)$, respectively. So,

$$\frac{\partial \mathcal{H}}{\partial P(y|x)} = P(x) \log P(x|y) - \mathrm{T}P(x) \log \frac{P(y|x)}{P(y)} - \mathrm{T}P(x) + \lambda_x.$$

An equation $\partial \mathcal{H} / \partial P(y|x) = 0$ leads to

$$\mathrm{T} \log \frac{P(y|x)}{P(y)} = \frac{\lambda_x}{P(x)} - \mathrm{T} + \log P(x|y) = \lambda'_x + \log P(x|y),$$

and it follows that $P(y|x)/P(y) = e^{\lambda'_x/\mathrm{T}} P(x|y)^{1/\mathrm{T}}$, which means $P(y|x) = \lambda''_x P(y) P(x|y)^{1/\mathrm{T}}$. Because λ''_x is determined so that $\sum_y P(y|x) = 1$, Equation (4.49) is obtained.

[23] This computation corresponds to E-step (note that Equation (4.49) equals to Equation (4.18) if $\mathrm{T} = 1$).

[24] In literature, Equation (4.50) is often referred to as *hard assignment*, whereas $P(x|y)$ (i.e., $P(\mathbf{d}_i|C_k)$) is directly used in the case of *soft assignment*. Actually, the vMF distribution was used as the model in [18].

4.1.2 Bayesian model of multinomial mixture

Basic Bayesian formula

From Equations (4.3), (4.8) and (4.28), the multinomial mixture model (MMM) can be again written as

$$P(\mathbf{y};\mathbf{\Psi}) = \prod_{i=1}^{N} P(\mathbf{d}_i;\mathbf{\Psi}) = \prod_{i=1}^{N}\sum_{k=1}^{L} \eta_k A_i \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}}, \tag{4.51}$$

where $\mathbf{y} = [\mathbf{d}_1^T, \ldots, \mathbf{d}_N^T]^T$ and $\mathbf{\Psi}$ denotes a vector of parameters as before. In the Bayesian approach, a probability of parameter vector $\mathbf{\Psi}$ and a conditional probability of observed data $\mathbf{y}$ given the parameter vector $\mathbf{\Psi}$ are explicitly introduced. As a result, based on the Bayes' theorem, a *posterior* probability distribution $P(\mathbf{\Psi}|\mathbf{y})$ is computed such that

$$P(\mathbf{\Psi}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{\Psi})P(\mathbf{\Psi})}{P(\mathbf{y})}, \tag{4.52}$$

where

$$P(\mathbf{y}) = \int P(\mathbf{y}|\mathbf{\Psi})P(\mathbf{\Psi})d\mathbf{\Psi}, \tag{4.53}$$

if $P(\mathbf{\Psi})$ is assumed to be a probability density function of $\mathbf{\Psi}$ [25]. In these equations, $P(\mathbf{y}|\mathbf{\Psi})$ is the 'observational' distribution, and posterior $P(\mathbf{\Psi}|\mathbf{y})$ can be interpreted as $\mathcal{L}(\mathbf{\Psi})$ in non-Bayesian manner [26]. Note that, since $P(\mathbf{y})$ is constant over all $\mathbf{\Psi}$s, Equation (4.52) is often treated as

$$P(\mathbf{\Psi}|\mathbf{y}) \propto P(\mathbf{y}|\mathbf{\Psi})P(\mathbf{\Psi}). \tag{4.54}$$

Usually, $P(\mathbf{\Psi})$ is called *prior* probability distribution, which represents 'knowledge' on the parameters held before observation. The prior knowledge will be updated into posterior distribution $P(\mathbf{\Psi}|\mathbf{y})$ by using observational distribution $P(\mathbf{y}|\mathbf{\Psi})$ through Equation (4.52). This means that the Bayesian approach tries to estimate basically distribution $P(\mathbf{\Psi}|\mathbf{y})$, not to calculate a point or interval estimate of $\mathbf{\Psi}$, which makes often *Bayesian inference* more complicated than conventional statistical inference.

Since the mixing and multinomial parameters are independent each other in the case of the MMM, the prior becomes

$$P(\mathbf{\Psi}) = P(\boldsymbol{\eta}, \mathbf{p}) = P(\boldsymbol{\eta})P(\mathbf{p}) = P(\boldsymbol{\eta})P(\mathbf{p}_1)\ldots P(\mathbf{p}_L), \tag{4.55}$$

where $\boldsymbol{\eta} = [\eta_1, \ldots, \eta_L]^T$, $\mathbf{p} = [\mathbf{p}_1^T, \ldots, \mathbf{p}_L^T]^T$ and $\mathbf{p}_k = [p_{1|k}, \ldots, p_{M|k}]^T$ $(k = 1, \ldots, L)$, and *Dirichlet distribution* is often used as $P(\cdot)$ in Equation (4.55). For instance, the Dirichlet distribution of $\mathbf{p}_k$ $(k = 1, \ldots, L)$ can be written as

$$P(\mathbf{p}_k|\boldsymbol{\beta}) = \frac{\Gamma(\sum_{j=1}^{M}\beta_j)}{\prod_{j=1}^{M}\Gamma(\beta_j)} \prod_{j=1}^{M} p_{j|k}^{\beta_j - 1}, \tag{4.56}$$

where $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_M]^T$, which is a vector of parameters, and $\Gamma(\cdot)$ denotes a Gamma function. Note that Equation (4.56) is a probability density function as mentioned above. According to Bayesian convention, $P(\mathbf{p}_k)$ is sometimes denoted by $P(\mathbf{p}_k|\boldsymbol{\beta})$ in this chapter where each element of

[25] Strictly, for a continuous random variable x, the probability is computed as

$$P(x \leq a) = \int_{-\infty}^{a} f(x)dx,$$

where $f(x)$ means a probability density function. Therefore, $P(\mathbf{\Psi})$ should be properly written by using a notation other than $P(\cdot)$ (e.g., $f(\mathbf{\Psi})$). However, for simplicity, the notation $P(\cdot)$ is used for continuous random variables in this thesis because the parameters in $\mathbf{\Psi}$ will be always integrated out in the scope of discussions here. Namely, it is enough to notice that an integral like that in Equation (4.53) is valid for $P(\mathbf{\Psi})$ in this thesis. Also, an integral of random vector $\mathbf{x}$ is defined as $\int f(\mathbf{x})d\mathbf{x} = \int\int \ldots \int f(x_1, \ldots, x_n)dx_1dx_2\ldots dx_n$ where $\mathbf{x} = [x_1, \ldots, x_n]^T$. If $x_1, \ldots, x_n$ are independent each other, then $\int f(\mathbf{x})d\mathbf{x} = \int f(x_1)dx_1 \times \ldots \times \int f(x_n)dx_n$, which is always assumed in this section.

[26] Therefore, the maximum likelihood (ML) estimation corresponds to 'maximum a posterior' (MAP) estimation in the Bayesian framework.

Table 4.8: Example of Dirichlet probability density

$p_{1\|k}$	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$p_{2\|k}$	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1
$\beta_1 = \beta_2 = 5$	0.0	0.4	1.2	2.1	2.5	2.1	1.2	0.4	0.0
$\beta_1 = \beta_2 = 20$	0.0	0.0	0.2	2.3	5.0	2.3	0.2	0.0	0.0
$\beta_1 = \beta_2 = 50$	0.0	0.0	0.0	1.1	8.0	1.1	0.0	0.0	0.0
$\beta_1 = 2, \beta_2 = 8$	3.4	3.0	1.8	0.8	0.3	0.1	0.0	0.0	0.0
$\beta_1 = 10, \beta_2 = 40$	1.3	7.0	1.5	0.0	0.0	0.0	0.0	0.0	0.0

$\boldsymbol{\beta}$ is often called *hyperparameter*. Naturally, it is also possible to assume a probabilistic distribution $P(\boldsymbol{\beta})$ for the hyperparameter, but such complicated model is not discussed in this thesis.

A main reason that the Dirichlet distribution is used in MMM, it is *conjugate* to the multinomial distribution [99]. Namely, if $P(\mathbf{d}_i|\mathbf{p}_k)$ follows a multinomial distribution, then it becomes that

$$P(\mathbf{p}_k|\mathbf{d}_i) \propto P(\mathbf{d}_i|\mathbf{p}_k)P(\mathbf{p}_k) \propto \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}} \prod_{j=1}^{M} p_{j|k}^{\beta_j - 1} = \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij} + \beta_j - 1}, \tag{4.57}$$

which means that posteriori distribution $P(\mathbf{p}_k|\mathbf{d}_i)$ becomes again a Dirichlet distribution (note that '$\propto$' is used after $P(\mathbf{d}_i|\mathbf{p}_k)P(\mathbf{p}_k)$ because constants of the two distributions are omitted).

Table 4.8 exemplifies actual values of $P(\mathbf{p}_k|\boldsymbol{\beta})$ when $L = 2$. If the distribution is 'symmetric' (i.e., β_1 and β_2 take a same value), then the value is the maximum in the case that $p_{1|k} = p_{2|k} = 0.5$, and becomes larger as β_1 and β_2 increase. Meanwhile, when $\beta_1 = 10$ and $\beta_2 = 40$ (i.e., a non-symmetric), the value is the maximum in the case that $p_{1|k} = 10/50 = 0.2$ and $p_{2|k} = 40/50 = 0.8$ as shown in Table 4.8. Since $p_{j|k}$ $(j = 1, \ldots, M)$ is a discrete probability distribution for a fixed k, the Dirichlet distribution can be interpreted as a 'distribution of distribution' as shown in the table.

Consequently, the posterior distribution of MMM can be written as

$$\begin{aligned} P(\boldsymbol{\eta}, \mathbf{p}|\mathbf{y}) &\propto P(\mathbf{y}|\boldsymbol{\eta}, \mathbf{p})P(\boldsymbol{\eta})P(\mathbf{p}) \\ &\propto \prod_{i=1}^{N} \sum_{k=1}^{L} \eta_k \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}} \times \prod_{k=1}^{L} \eta_k^{\alpha_k - 1} \times \prod_{k=1}^{L} \prod_{j=1}^{M} p_{j|k}^{\beta_j - 1}, \end{aligned} \tag{4.58}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_L]^T$ is a hyperparameter vector of Dirichlet distribution $P(\boldsymbol{\eta}|\boldsymbol{\alpha})$ [251].

Estimation by EM algorithm

In order to estimate parameters in Equation (4.58) by an EM algorithm, unobservable variable z_{ik} indicating the cluster to which a document belongs has to be introduced again (see Equation (4.9)) so that Equation (4.58) becomes tractable. Actually, if the most right-hand side in Equation (4.58) is treated as $\mathcal{L}$, then the log likelihood for complete data can be written as

$$\begin{aligned} \log \mathcal{L}_c(\boldsymbol{\Psi}) &= \sum_{k=1}^{L} \sum_{i=1}^{N} z_{ik} \log \eta_k + \sum_{k=1}^{L} \sum_{i=1}^{N} z_{ik} \log \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}} \\ &\quad + \sum_{k=1}^{L} (\alpha_k - 1) \log \eta_k + \sum_{k=1}^{L} \sum_{j=1}^{M} (\beta_j - 1) \log p_{j|k}, \end{aligned} \tag{4.59}$$

which corresponds to Equation (4.10) in non-Bayesian case, and the objective function $\mathcal{H}$ in Equations (4.19) and (4.32) can be straightforwardly used. First, since updating formula of z_{ik} in E-step is the same with Equation (4.18),

$$z_{ik}^{(s)} = \frac{\eta_k^{(s)} \prod_j p_{j|k}^{(s)\mathrm{f}_{ij}}}{\sum_{k'=1}^{L} \eta_{k'}^{(s)} \prod_j p_{j|k'}^{(s)\mathrm{f}_{ij}}}, \tag{4.60}$$

Table 4.9: Result of experiment of Bayesian MMM (1000 runs with $L = 3$)

	$\alpha_k = 1.0$ $\beta_j = 1.0$		$\alpha_k = 2.0$ $\beta_j = 2.0$		$\alpha_k = 4.0$ $\beta_j = 4.0$		$\alpha_k = 6.0$ $\beta_j = 6.0$	
	$\log \mathcal{L}$	#	$\log \mathcal{L}$	#	$\log \mathcal{L}$	#	$\log \mathcal{L}$	#
1	-46.8	68	-52.9	442	-60.8	482	-66.1	1000
2	-47.2	266	-53.4	357	-61.3	331		
3	-47.3	153	-54.2	185	-62.4	187		
4	-47.6	106	-60.9	8				
5	-48.5	102	-63.0	8				
6	-48.8	32						
7	-49.7	3						
8	-50.7	18						
9	-51.6	67						
10	-51.8	23						
11	Others	162						
Total	-	1000	-	1000	-	1000	-	1000

Note: 'Others' contains 23 different local maximums.

can be obtained again. Second, by solving $\partial \mathcal{H}/\partial \eta_k = 0$ and $\partial \mathcal{H}/\partial p_{j|k} = 0$ based on Equations (4.19) and (4.32), respectively, it follows that

$$\eta_k^{(s+1)} = \frac{\alpha_k - 1 + \sum_i z_{ik}^{(s)}}{\sum_{k'} \left(\alpha_{k'} - 1 + \sum_i z_{ik'}^{(s)} \right)}, \tag{4.61}$$

and

$$p_{j|k}^{(s+1)} = \frac{\beta_j - 1 + \sum_i \mathrm{f}_{ij} z_{ik}^{(s)}}{\sum_{j'} \left(\beta_{j'} - 1 + \sum_i \mathrm{f}_{ij'} z_{ik}^{(s)} \right)}, \tag{4.62}$$

which are slightly changed from Equations (4.20) and (4.31) because the third and fourth terms in the right-hand side of Equation (4.59) are added [27]. It is clear that the Bayesian model of multinomial mixture reduces to the standard MMM if all hyperparameters are 1.0 (i.e., $\alpha_1 = \ldots = \alpha_L = 1.0$ and $\beta_1 = \ldots = \beta_M = 1.0$).

Table 4.9 shows the result of an experiment for the Bayesian model of multinomial mixture by using the sample DB when $L = 3$ [28]. As before, 1000 runs were repeated with varying initial values of parameters, in which only symmetric Dirichlet distributions with a common value for the hyperparameters were used. Therefore, the column of $\alpha_k = \beta_j = 1.0$ indicates a result of clustering based on the non-Bayesian MMM discussed before.

Whereas many local maximums were derived from executions by the EM algorithm of non-Bayesian (standard) MMM, it is clear from Table 4.9 that the local maximum problem is drastically improved in its Bayesian versions. Especially, runs based on the Bayesian model with $\alpha_k = \beta_j = 6.0$ always reached to the global maximum with $\log \mathcal{L} = -66.1$, which successfully provide a 'valid' set of clusters, $C_1 = \{d_1, d_2, d_3\}$, $C_2 = \{d_4, d_5, d_7\}$ and $C_3 = \{d_6, d_8, d_9, d_{10}\}$, in this experiment. The improvement was brought by a 'smoothing' effect of incorporating hyperparameters of the Dirichlet distribution into the model. For example, Table 4.10 shows estimation of $p_{j|k}$ by the Bayesian MMM with $\alpha_k = \beta_j = 6.0$. By comparing it with Table 4.3 of the non-Bayesian MMM, it turns out that the mass of probability given to each cluster is more widely spread over terms. In

[27] For instance, differential of $\sum_k (\alpha_k - 1) \log \eta_k$ with respect to η_k is calculated as $(\alpha_k - 1)/\eta_k$, which is added to the original Equation (4.20).

[28] Note that the log likelihood values in Table 4.9 were computed as

$$\log \mathcal{L} = \log \left[\prod_{i=1}^{N} A_i \sum_{k=1}^{L} \eta_k \prod_{j=1}^{M} p_{j|k}^{\mathrm{f}_{ij}} \right],$$

by using estimations of η_k and $p_{j|k}$ based on Equation (4.58).

Table 4.10: Values of $p_{j|k}$ by Bayesian MMM

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$k=1$	0.226	0.324	0.122	0.106	0.115	0.106
$k=2$	0.105	0.147	0.301	0.171	0.144	0.153
$k=3$	0.092	0.114	0.116	0.191	0.335	0.133

Note: $\log L(\mathbf{\Psi}) = -66.1$, and $\alpha_k = \beta_j = 6.0$.

other words, the differences of probability between terms become smaller in the case of Bayesian MMM, which was caused by adding a constant 5.0 ($= \beta_k - 1.0$) to the numerator of Equation (4.62) for all terms. The addition of hyperparameters common to all terms and clusters contributes to avoiding 'over-fitting' in the estimation, which would lead to reduction of local maximums. As this example shows, hyperparameters in the Bayesian MMM play an important role for obtaining good clusters with avoiding to reach to a local maximum.

Estimation by Gibbs sampling

If unobservable variable vector $\mathbf{z}$, which indicates allocation of each document to a cluster, is assumed to be completely known, then the observational distribution of the Bayesian model of multinomial mixture can be written as

$$P(\mathbf{y}|\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}) = \prod_{i=1}^{N} A_i \times \prod_{k=1}^{L} \eta_k^{\tilde{n}_k} \prod_{j=1}^{M} p_{j|k}^{\tilde{\mathrm{f}}_{kj}}, \tag{4.63}$$

where

$$\tilde{\mathrm{f}}_{kj} = \sum_{i:d_i \in C_k} \mathrm{f}_{ij}, \tag{4.64}$$

which means the total frequency of occurrences of term t_j in cluster C_k, and $\tilde{n}_k$ denotes the number of documents allocated to cluster C_k (i.e., $|C_k| = \tilde{n}_k$). Therefore, the posterior distribution becomes

$$P(\boldsymbol{\eta}, \mathbf{p}|\mathbf{y}, \mathbf{z}) \propto P(\mathbf{y}|\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}) P(\boldsymbol{\eta}) P(\mathbf{p}) \propto \prod_{k=1}^{L} \eta_k^{\tilde{n}_k + \alpha_k - 1} \prod_{j=1}^{M} p_{j|k}^{\tilde{\mathrm{f}}_{kj} + \beta_j - 1} \tag{4.65}$$

after the complete data $\mathbf{x} = [\mathbf{y}^T, \mathbf{z}^T]^T$ are observed.

Actually, $\mathbf{z}$ can not be observed. Thus an EM algorithm for the Bayesian MMM tries to estimate $\boldsymbol{\eta}$ and $\mathbf{p}$ based on a criterion of maximizing analytically Equation (4.65) by employing tentatively values of $\mathbf{z}$ estimated in the previous step of iterations. In the algorithm, random numbers are used only for assigning initial values to $\boldsymbol{\eta}$ and $\mathbf{p}$, and the same result of estimation is always obtained for the same initial values. In this sense, the EM algorithm can be considered as a 'deterministic' approach. Meanwhile, a 'stochastic' technique for estimating distribution $P(\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}|\mathbf{y})$, in which $\mathbf{z}$ is incorporated into the set of random variables on the left-hand side, is also feasible for estimating parameters as a computational simulation. Because $\mathbf{z}$ is explicitly treated as a parameter vector to be estimated, not as unobservable data, the set of parameters can be rewritten as $\mathbf{\Psi} = [\mathbf{z}^T, \boldsymbol{\eta}^T, \mathbf{p}^T]^T$.

Generally, if a simulation method is applied to 'imitate' probabilistic distribution $P(x)$, then the computer generates a series of actual values of x so that its frequency distribution is approximately equivalent to $P(x)$. When x is a discrete variable, a practical strategy is to adopt the mode (i.e., a value appearing most frequently) as an estimate of x because $P(x)$ is empirically maximized at the mode of x. Thus if frequency data on $P(\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}|\mathbf{y})$ are obtained by a stochastic simulation, then it is possible to partition the target document set into clusters based on the empirical data of $\mathbf{z}$.

Among several simulation methods for generating such frequency data, *Gibbs sampling* method, which is a specific version of *Markov chain Monte Carlo* (MCMC) [99, 252], is often used for estimating parameters of Bayesian models. It is assumed that an m-dimensional parameter vector

$\boldsymbol{\Psi}$ has to be estimated in a situation that its joint probability $P(\Psi_1, \ldots, \Psi_m|\mathbf{y})$ is intractable but it is relatively easy to calculate analytically a conditional probability

$$P(\Psi_h|\mathbf{y}; \Psi_1, \ldots, \Psi_{h-1}, \Psi_{h+1}, \ldots, \Psi_m). \tag{4.66}$$

This is often called 'full conditional' probability. The Gibbs sampling method tries to generate actual values of $\boldsymbol{\Psi}$ following $P(\boldsymbol{\Psi}|\mathbf{y})$ as a Markov chain by using the full conditional probability. Thus even if $P(\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}|\mathbf{y})$ has a complicated form, it is possible to simulate $P(\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}|\mathbf{y})$ when Equation (4.66) can be analytically computed. Whether the full conditional probability can be obtained or not is a critical point in applying the Gibbs sampling method.

More specifically, after selecting randomly initial values $\Psi_1^{(0)}, \ldots, \Psi_m^{(0)}$, samples $\boldsymbol{\Psi}^{(s+1)}$ $(s = 0, 1, \ldots, r-1)$ are repeatedly drawn from each current conditional probability distribution in Equation (4.66) based on random numbers generated by a computer as follows [208]:

1. Draw $\Psi_1^{(s+1)}$ from $P(\Psi_1|\mathbf{y}; \Psi_2^{(s)}, \ldots, \Psi_m^{(s)})$.
2. Draw $\Psi_2^{(s+1)}$ from $P(\Psi_2|\mathbf{y}; \Psi_1^{(s+1)}, \Psi_3^{(s)}, \ldots, \Psi_m^{(s)})$.

   ...

   m. Draw $\Psi_m^{(s+1)}$ from $P(\Psi_m|\mathbf{y}; \Psi_1^{(s+1)}, \ldots, \Psi_{m-1}^{(s+1)})$.

A series of drawing Ψ_v $(v = 1, \ldots, m)$ constitutes a Markov chain in which values of each Ψ_v appear following $P(\boldsymbol{\Psi}|\mathbf{y})$ under a condition [29]. Usually, some beginning samples (e.g., $\{\Psi_v^{(s)}|s = 1, \ldots, B\}$ where B is a number) are discarded because they may have large effect from initial values. The earlier stage is often called 'burn-in period'. If samples after the burn-in-period are recorded, then parameter Ψ_v can be estimated by detecting the most frequent value. When Ψ_v is a continuous variable, it is possible to estimate the parameter such that $\hat{\Psi}_v = (r-B)^{-1} \sum_{s=B+1}^{r} \Psi_v^{(s)}$, which is the mean of an empirical distribution of Ψ_v produced from the samples. Furthermore, other statistics such as confidential intervals can be easily obtained from the empirical distribution. This is an advantage of the Gibbs sampling in comparison with EM algorithms (fortunately, simple DC does not has to use such statistics).

In the case of Gibbs sampling for the Bayesian MMM, it is necessary to determine three conditional distributions, $P(\mathbf{z}|\boldsymbol{\eta}, \mathbf{p}, \mathbf{y})$, $P(\boldsymbol{\eta}|\mathbf{p}, \mathbf{z}, \mathbf{y})$ and $P(\mathbf{p}|\boldsymbol{\eta}, \mathbf{z}, \mathbf{y})$. First, it is reasonable to use Equation (4.60) as a discrete distribution $\pi(k)$ $(k = 1, \ldots, L)$ for sampling from $P(\mathbf{z}|\boldsymbol{\eta}, \mathbf{p}, \mathbf{y})$ [251]. More specifically, a discrete distribution $\pi_i(k)$ for particular d_i, which provides probabilities for drawing index k of a cluster from $\{1, \ldots, L\}$, becomes

$$\pi_i(k) = P(z_{ik} = 1|\boldsymbol{\eta}, \mathbf{p}, \mathbf{y}) = \frac{\eta_k^{(s)} \prod_j p_{j|k}^{(s)\mathrm{f}_{ij}}}{\sum_{k'=1}^{L} \eta_{k'}^{(s)} \prod_j p_{j|k'}^{(s)\mathrm{f}_{ij}}}. \tag{4.67}$$

In actual sampling, a cumulative distribution is constructed such that $F_i(1) = \pi_i(1)$, $F_i(2) = F_i(1) + \pi_i(2)$, $\ldots, F_i(L-1) = F_i(L-2) + \pi_i(L-1), F_i(L) = 1.0$. For example, if $\pi_i(1) = 0.3$, $\pi_i(2) = 0.4$, $\pi_i(3) = 0.3$ and $L = 3$, then $F_i(1) = 0.3$, $F_i(2) = 0.7$ and $F_i(3) = 1.0$. After that, a random number U following a uniform distribution whose range is from 0 to 1 (which is denoted by $U \sim \mathcal{U}[0, 1]$) is generated by a computer program (e.g., Math.random() in Java), and z_{ik} is determined as

$$z_{ik}^{(s+1)} = \begin{cases} 1 & \text{if } F_i(k-1) < U \leq F_i(k) \\ 0 & \text{otherwise} \end{cases}, \; k = 1, \ldots, L, \tag{4.68}$$

which means that a 'new' allocation of document d_i to a cluster is drawn as a sample at $(s+1)$-th iteration [30].

By using the sampling result, it is possible to compute the number of documents and total frequency of term occurrences in each cluster at $(s+1)$-th iteration such that

$$\tilde{n}_k^{(s+1)} = \sum_{i=1}^{N} z_{ik}^{(s+1)}, \tag{4.69}$$

[29] Roughly speaking, by selecting repeatedly a value of x_i according to $P(x_i|x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ for $i = 1, \ldots, n$, various 'states' of $\mathbf{x} = [x_1, \ldots, x_n]^T$ appear in a chain of the samples. If the chain is *ergodic*, which means that it reaches to all states, then an empirical distribution of $P(\mathbf{x})$ can be produced by counting each state of $\mathbf{x}$ in an enough number of samples.

[30] In this notation, since $F_i(0)$ is not defined, $z_{i1}^{(s+1)} = 1$ if $U \leq F_i(1)$. See [252] for details on implementing practically the generation of random numbers following a discrete distribution.

and

$$\tilde{\mathrm{f}}_{kj}^{(s+1)} = \sum_{i=1}^{N} z_{ik}^{(s+1)} \mathrm{f}_{ij}, \tag{4.70}$$

for $k = 1, \ldots, L$. These values will be used for sampling from $P(\boldsymbol{\eta}|\mathbf{p}, \mathbf{z}, \mathbf{y})$ and $P(\mathbf{p}|\boldsymbol{\eta}, \mathbf{z}, \mathbf{y})$ in the step. Since $\boldsymbol{\eta}$, $\mathbf{p}_1$, ..., $\mathbf{p}_L$ are all independent each other, it is clear from Equation (4.65) that $\boldsymbol{\eta}^{(s+1)}$ should be drawn from a Dirichlet distribution with parameters

$$\tilde{\boldsymbol{\alpha}} = [\tilde{n}_1^{(s+1)} + \alpha_1, \ldots, \tilde{n}_L^{(s+1)} + \alpha_L]^T \tag{4.71}$$

(see [263]) [31]. Similarly, a sample of $\mathbf{p}_k^{(s+1)}$ can be obtained from the Dirichlet distribution with parameters

$$\tilde{\boldsymbol{\beta}}_k = [\tilde{\mathrm{f}}_{k1}^{(s+1)} + \beta_1, \ldots, \tilde{\mathrm{f}}_{kM}^{(s+1)} + \beta_M]^T, \tag{4.72}$$

$(k = 1, \ldots, L)$.

After s is incremented (i.e., $s \leftarrow s + 1$), a new iteration is started from Equation (4.67). By repeating the iterative procedure (e.g., 1000 times), empirical distributions of $P(\mathbf{z}|\boldsymbol{\eta}, \mathbf{p}, \mathbf{y})$, $P(\boldsymbol{\eta}|\mathbf{z}, \mathbf{y})$ and $P(\mathbf{p}|\mathbf{z}, \mathbf{y})$ can be produced from a set of samples. Figure 4.7 shows the procedure of Gibbs sampling for the MMM. In order to create a cluster set from the samples, it is enough to count the number of allocations in the set of samples, and to determine the final clusters as

$$C_k = \left\{ d_i \,\middle|\, k = \underset{k'=1,\ldots,L}{\arg\max} \sum_{s=B+1}^{r} z_{ik'}^{(s)} \right\}, \tag{4.73}$$

which corresponds to finding the mode of allocations for each document.

‡ Clustering by MMM based on Gibbs sampling

Set: The number of clusters L, the Dirichlet parameters α_k and β_j $(k = 1, \ldots, L;\ j = 1, \ldots, M)$, the number of total iterations r and constant B in Equation (4.73).

1) Initialize parameters $\eta_1, \ldots, \eta_L$ and $p_{1|1}, \ldots, p_{M|L}$ by generating random numbers, and set s to 1 (i.e., $s \leftarrow 1$).

2) For $i = 1, \ldots, N$, compute Equation (4.67) and allocate d_i randomly to a cluster according to Equation (4.68).

3) Based on resulting allocations in step 2), compute Equations (4.69) and (4.70) for $k = 1, \ldots, L$ and $j = 1, \ldots, M$.

4) Draw s-th samples of η_k and $p_{j|k}$ from the Dirichlet distributions with parameters in Equations (4.71) and (4.72), respectively, for $k = 1, \ldots, L$ and $j = 1, \ldots, M$.

5) $s \leftarrow s + 1$. If $s > r$, then go to step 6). Otherwise, return to 2).

6) Terminate the procedure after allocating d_i to a cluster based on Equation (4.73) for $i = 1, \ldots, N$.

Out: Clusters $C_1, \ldots, C_L$ and $r - B$ samples of parameters η_k and $p_{j|k}$.

Figure 4.7: Clustering by MMM based on Gibbs sampling

Actually, in order to draw samples from the Dirichlet distributions at the step 4) in Figure 4.7, it is necessary to generate random numbers following the Dirichlet distributions with parameters in Equations (4.71) and (4.72). The generation of random numbers following various distributions is an essential part of the stochastic simulation like MCMC [32]. If random numbers X_i $(i = 1, \ldots, \alpha)$

[31] Due to the independence assumption, $P(\boldsymbol{\eta}, \mathbf{p}|\mathbf{y}, \mathbf{z}) = P(\boldsymbol{\eta}|\mathbf{y}, \mathbf{z})P(\mathbf{p}|\mathbf{y}, \mathbf{z})$ and $P(\boldsymbol{\eta}|\mathbf{p}, \mathbf{y}, \mathbf{z}) = P(\boldsymbol{\eta}|\mathbf{y}, \mathbf{z})$. Therefore, $P(\boldsymbol{\eta}|\mathbf{p}, \mathbf{y}, \mathbf{z})$ is only dependent on term $\prod_k \eta_k^{\tilde{n}_k + \alpha_k - 1}$ in Equation (4.65), which leads naturally to sampling based on the Dirichlet distribution.

[32] Various sophisticated techniques for the generation of random variables have been developed (e.g., see [252]).

follow an Exponential distribution $F(x) = 1 - e^{-x}$ (where $F(x)$ represents the cumulative probability distribution), then $\tilde{G} = \sum_{i=1}^{\alpha} X_i$ is distributed as a Gamma distribution with parameters α and 1. Suppose that there are L random numbers following Gamma distributions with parameters α_k and 1, respectively ($k = 1, \ldots, L$), each of which is denoted by $\tilde{G}_k$. Then, a random variable vector $\mathbf{Y} = [Y_1, \ldots, Y_L]^T$ following a Dirichlet distribution with parameters $\alpha_1, \ldots, \alpha_L$ is computed as $Y_k = \tilde{G}_k / \sum_{k'=1}^{L} \tilde{G}_{k'}$ for $k = 1, \ldots, L$ (e.g., see [99]).

A random number of the Exponential distribution at its staring point can be easily generated by using a random number following a uniform $\mathcal{U}[0, 1]$ according to a relation $x = -\log(1 - u)$ where x is a random variable of the Exponential distribution and u is of $\mathcal{U}[0, 1]$ [33]. Consequently, random numbers following the Dirichlet distribution can be generated from random numbers of $\mathcal{U}[0, 1]$ [34] similarly to Equation (4.68).

Unfortunately, when applying the Gibbs sampling method to a mixture model, it is necessary to solve a so-called *label-switching* problem (see p.129 in [209]) because the criterion of maximizing the likelihood $P(\boldsymbol{\eta}, \mathbf{p}, \mathbf{z}|\mathbf{y})$ does not specify a unique order of mixture components. For example, the likelihood score of set $C_1 = \{d_1, d_2\}$ and $C_2 = \{d_3\}$ is completely equal to that of set $C_1 = \{d_3\}$ and $C_2 = \{d_1, d_2\}$, which means that the likelihood is invariant when labels switch between clusters. Although this problem is also caused in execution of the EM algorithm, the Gibbs sampling method suffers more serious damage from it because the estimation is based on cumulated samples.

The left-hand sides of Table 4.11 and Figure 4.8 indicate results from an execution of Gibbs sampling in Figure 4.7 for the sample DB when $L = 3$, $r = 1100$, $B = 100$ and $\alpha_k = \beta_j = 1.0$ for all k and j (which means that no prior knowledge on the parameters η_k and $p_{j|k}$ was incorporated into the Bayesian model) [35]. The table lists the numbers of allocations to clusters by each document and the figure shows a frequency distribution of samples of $p_{3|2}$ for t_3 in C_2. During the 1100 iterations, the label switching between C_2 and C_3 occurred frequently, which mixed up allocations to the two clusters in 1000 iterations after the burn-in period. For example, whereas

• 397th iteration: $C_2 = \{d_6, d_8, d_9, d_{10}\}$ and $C_3 = \{d_4, d_5, d_7\}$

provided a 'valid' set of clusters, d_6 moved 'accidentally' to C_3 by a random allocation in next 398th iteration. After eight iterations, the 'valid' set of clusters came back again as

• 406th iteration: $C_2 = \{d_4, d_5, d_7\}$ and $C_3 = \{d_6, d_8, d_9, d_{10}\}$

in which labels of clusters were exchanged. Such mixing of two clusters throughout all iterations brought an invalid result as shown in the left-hand sides of Table 4.11 and Figure 4.8.

In general, various techniques for alleviating the label switching problem have been proposed (e.g., see [263]). A simple 'batch mode' method would be to change operationally the labels based on values of $p_{j|k}$ at each iteration after obtaining all allocations from $r - B$ iterations. More specifically, by using a particular iteration achieving the largest log likelihood as a criterion, cluster labels of each iteration can be heuristically changed according to 'closeness' of its $\mathbf{p}^{(s)}$ to that of the criterion. The estimate of $\mathbf{p}$ at the iteration with the largest log likelihood is denoted by $\tilde{\mathbf{p}} = [\tilde{p}_{1|1}, \ldots, \tilde{p}_{M|L}]^T$ here. For example, in the experiment using the sample DB, the 559th iteration provided the largest log likelihood and its allocation of documents was $C_2 = \{d_4, d_5, d_7\}$ and $C_3 = \{d_6, d_8, d_9, d_{10}\}$, cluster labels of which are different from those at the 397th iteration mentioned above. Intuitively, it is expected that $\mathbf{p}$ at the 397th iteration becomes closer to $\tilde{\mathbf{p}}$ if cluster numbers 2 and 3 are swapped.

In order to detect automatically effective swapping of labels, it is necessary to check all $L!$ permutations of $k = 1, \ldots, L$, which are denoted by $\sigma(k; b)$ ($b = 1, \ldots, L!$). In the case of 397th iteration, if $\sigma(1; 1) = 1$, $\sigma(2; 1) = 3$ and $\sigma(3; 1) = 2$ when $b = 1$, then this permutation would provide the closest $\mathbf{p}$ to $\tilde{\mathbf{p}}$ among all permutations. The closeness between $\mathbf{p}$ of each iteration and

[33] Like the case of discrete distributions in Equation (4.68), random numbers of a continuous variable X following the target probability distribution (e.g, Exponential) can be generated by corresponding its cumulative function $F(X)$ to U ($\sim \mathcal{U}[0, 1]$). When putting that $U = F(X)$, $P(U \leq u) = P[F(X) \leq F(x)]$ for a pair of u and x. If there is an inverse function $x = F^{-1}[F(x)]$, then $P[F(X) \leq F(x)] = P[F^{-1}(F(X)) \leq F^{-1}(F(x))] = P(X \leq x)$, which means that $x = F^{-1}(u)$ follows the target distribution. Since $u = F(x) = 1 - e^{-x}$ in this case of Exponential distribution, it follows that $x = F^{-1}(u) = -\log(1 - u)$ [252].

[34] The statistical language R [246] provides rgamma() function generating directly a random number of the Gamma distribution. The other sophisticated algorithms of generating Gamma random numbers are also available (e.g., [299]).

[35] When the procedure is executed only once, it brings a 'single chain'. In contrast, if the procedure is repeated with changing initial parameters, then multiple chains are implemented. The execution of this experiment is a single chain.

Table 4.11: Allocations of documents to clusters in sample DB by Gibbs sampling

	Before changing labels				After changing labels			
	C_1	C_2	C_3	Total	C_1	C_2	C_3	Total
d_1	975	12	13	1000	980	13	7	1000
d_2	979	9	12	1000	983	12	5	1000
d_3	992	0	8	1000	996	2	2	1000
d_4	0	512	488	1000	4	843	153	1000
d_5	143	404	453	1000	147	642	211	1000
d_6	11	406	583	1000	14	335	651	1000
d_7	7	512	481	1000	9	879	112	1000
d_8	6	459	535	1000	8	218	774	1000
d_9	2	462	536	1000	5	213	782	1000
d_{10}	5	530	465	1000	4	343	653	1000

Note: $\alpha_k = \beta_j = 1.0$, $r = 1100$ and $B = 100$.

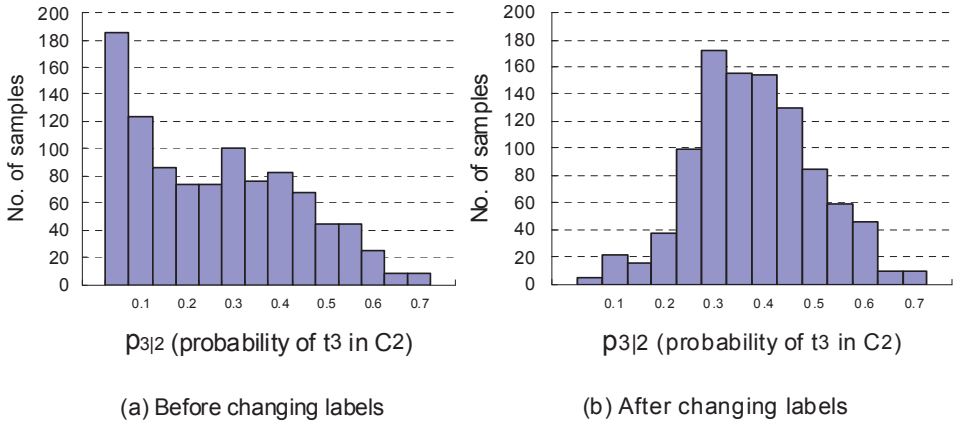


Figure 4.8: Distribution of samples on $p_{3|2}$ for sample DB

Table 4.12: Averages of $\hat{p}_{j|k}$ after changing cluster labels in Gibbs sampling

	$j=1$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$
$k=1$	0.307	0.499	0.053	0.043	0.053	0.044
$k=2$	0.045	0.120	0.354	0.176	0.183	0.121
$k=3$	0.044	0.078	0.102	0.210	0.408	0.157

Note: $\alpha_k = \beta_j = 1.0$, $r = 1100$ and $B = 100$.

$\tilde{\mathbf{p}}$ can be measured by the well-known Kullback-Leibler divergence such that

$$\mathcal{D}_{KL}(\tilde{\mathbf{p}} \| \mathbf{p}^{(s)}; b) = \sum_{k=1}^{L} \sum_{j=1}^{M} \tilde{p}_{j|k} \log \frac{\tilde{p}_{j|k}}{p^{(s)}_{j|\sigma(k;b)}}, \tag{4.74}$$

which leads to changing cluster labels of s-th iteration by using a permutation, $\sigma(1; b'), \ldots, \sigma(L; b')$ such that

$$b' = \underset{b=1,\ldots,L!}{\arg\min}\, \mathcal{D}_{KL}(\tilde{\mathbf{p}} \| \mathbf{p}^{(s)}; b), \tag{4.75}$$

for $s = B+1, \ldots, r$. Namely, the original label k is replaced by $\sigma(k; b')$ for each iteration in the storage of samples as a batch process ($k = 1, \ldots, L$).

The right-hand sides of Table 4.11 and Figure 4.8 show results after changing cluster labels based on Equation (4.75). It seems that both the clustering result and parameter estimation were drastically improved. Table 4.12 indicates also averages of $p_{j|k}$ after changing cluster labels, which are slightly different from estimations by the EM algorithm in Table 4.3.

A weak point of the MMM based on Gibbs sampling is its computational complexity. Clearly, many iterations are needed for obtaining a stable result even though only a single chain is executed. Additionally, for exchanging cluster labels, $\mathcal{D}_{KL}(\tilde{\mathbf{p}} \| \mathbf{p}^{(s)}; b)$ has to be computed $L!$ times for each iteration. In the case of DC, the number of clusters L is usually not small, and $L!$ computations may be infeasible.

However, if disregarding estimation of $p_{j|k}$, and trying only generation of document clusters, then it is possible to obtain easily clusters by counting 'cluster patterns' regardless of cluster numbers. For example, both the above 397th and 406th samples can be considered to provide the same set of clusters $\{d_1, d_2, d_3\}$, $\{d_4, d_5, d_7\}$ and $\{d_6, d_8, d_9, d_{10}\}$ even though labels of the latter two sets were different. From the purpose of DC, it would not be necessary to incorporate a complicated technique for solving the label switching problem, and it is enough to count occurrences of each cluster pattern in samples obtained after the burn-in period (the techniques is called 'cluster pattern count method' in this thesis for convenience).

4.1.3 Probabilistic latent semantic analysis (PLSA)

Latent classes of words

Probabilistic latent semantic analysis (PLSA) [36] proposed by Hofmann(1999) [130] is based on another framework for modeling probability of j-th term in i-th document (i.e., $P(t_j|d_i)$), in which unobservable variables τ_k $(k = 1, \ldots, L)$ are incorporated into calculation of the probability such that,

$$P(t_j|d_i) = \sum_{k=1}^{L} P(t_j|\tau_k) P(\tau_k|d_i). \tag{4.76}$$

Each τ_k can be interpreted as a *latent class* of words, which corresponds to a 'topic' discussed in documents. For example, when values of $P(\text{"family"}|\tau_k)$, $P(\text{"home"}|\tau_k)$ and $P(\text{"kid"}|\tau_k)$ are relatively larger, then variable τ_k may be postulated to represent a latent topic on 'family'.

Equation (4.76) can be considered as a *probabilistic topic model*, which is applicable to various applications of IR, text categorization or DC. In the case of DC, clusters C_k can be produced such

[36] PLSA is often called PLSI (probabilistic latent semantic indexing) or *aspect model*.

that

$$C_k = \left\{ d_i \,\middle|\, k = \underset{k'=1,\ldots,L}{\arg\max}\ P(\tau_{k'}|d_i) \right\}, \tag{4.77}$$

based on probabilities $P(\tau_k|d_i)$ $(k = 1, \ldots, L)$ [233].

Estimation in PLSA

For estimating $P(t_j|\tau_k)$ and $P(\tau_k|d_i)$ in PLSA from observed data, an EM algorithm is available. If $P(t_j|d_i)$ is assumed to be a parameter of multinomial distribution describing $P(\mathbf{d}_i)$, then the likelihood function given data $\mathbf{y} = [\mathbf{d}_1^T, \ldots, \mathbf{d}_N^T]^T$ becomes

$$\mathcal{L}(\mathbf{\Psi}) = \prod_{i=1}^{N} P(\mathbf{d}_i; \mathbf{\Psi}) = \prod_{i=1}^{N} A_i \prod_{j=1}^{M} P(t_j|d_i)^{\mathrm{f}_{ij}}, \tag{4.78}$$

where A_i is a multinomial coefficient of d_i (see Equation (4.28)). Therefore, by substituting Equation (4.76) for $P(t_j|d_i)$ in it, the log likelihood can be written as

$$\log \mathcal{L}(\mathbf{\Psi}) = A + \sum_{i=1}^{N} \sum_{j=1}^{M} \mathrm{f}_{ij} \log \sum_{k=1}^{L} P(t_j|\tau_k) P(\tau_k|d_i), \tag{4.79}$$

where $A = \sum_i \log A_i$ that does not contribute to estimation of parameters [37]. Therefore, unknown parameters in PLSA become

$$\mathbf{\Psi} = [P(t_1|\tau_1), \ldots, P(t_M|\tau_L), P(\tau_1|d_1), \ldots, P(\tau_L|d_N)]^T,$$

and the number of parameters amounts to $M \times L + L \times N = L \times (M + N)$.

In order to estimate the parameters by an EM algorithm, unobservable variables $z_{k|ij}$ $(k = 1, \ldots, L;\ i = 1, \ldots, N;\ j = 1, \ldots, M)$ are introduced where $z_{k|ij} = 1$ if a given pair of t_j and d_i is generated from τ_k, and otherwise, $z_{k|ij} = 0$. Suppose that every pair of a term and a document is generated from only a single latent topic. Under the assumption, if all $z_{k|ij}$ are observed (i.e., a particular τ_k is found for a given pair of t_j and d_i), then the log likelihood for complete data can be written as

$$\begin{aligned} \log L_c(\mathbf{\Psi}) &= A + \sum_{j=1}^{M} \sum_{i=1}^{N} \mathrm{f}_{ij} \log \sum_{k=1}^{L} z_{k|ij} P(t_j|\tau_k) P(\tau_k|d_i) \\ &= A + \sum_{j=1}^{M} \sum_{i=1}^{N} \mathrm{f}_{ij} \sum_{k=1}^{L} z_{k|ij} \log[P(t_j|\tau_k) P(\tau_k|d_i)]. \end{aligned} \tag{4.80}$$

In E-step, it is feasible to compute $\mathcal{Q}(\mathbf{\Psi}; \mathbf{\Psi}^{(s)})$ by using the same way of deriving Equation (4.18) because Equation (4.80) is a linear function and $z_{k|ij}$ is a zero-one indicator variable. Namely, an updating formula of unobservable variable $z_{k|ij}$ in E-step becomes

$$z_{k|ij}^{(s)} = P_{\mathbf{\Psi}^{(s)}}(Z_{k|ij} = 1|\mathbf{y}), \tag{4.81}$$

where $Z_{k|ij}$ is a random variable corresponding to $z_{k|ij}$ [38]. In the framework of PLSA, it is clearly reasonable to define that

$$P_{\mathbf{\Psi}}(Z_{k|ij} = 1|\mathbf{y}) = \frac{P(t_j|\tau_k) P(\tau_k|d_i)}{P(t_j|d_i)}, \tag{4.82}$$

[37] Originally, Hofmann(1999) [130] used $P(t_j, d_i)$, not $P(t_j|d_i)$, such that

$$P(d_i, t_j) = P(d_i) \sum_{k=1}^{L} P(t_j|\tau_k) P(\tau_k|d_i) = \sum_{k=1}^{L} P(\tau_k) P(t_j|\tau_k) P(d_i|\tau_k),$$

were $P(\tau_k)$, $P(t_j|\tau_k)$, and $P(d_i|\tau_k)$ were estimated by an EM algorithm.

[38] If putting that $g(z_{k|ij}) = z_{k|ij} \log[P(t_j|\tau_k) P(\tau_k|d_i)]$, then calculation similar to that in Equation(4.17) can be again repeated, and it turns out that $\mathcal{Q}(\mathbf{\Psi}; \mathbf{\Psi}^{(s)})$ is obtained by imputing $P_{\mathbf{\Psi}^{(s)}}(Z_{k|ij} = 1|\mathbf{y})$ into the position of $z_{k|ij}$, straightforwardly, in Equation (4.80).

which leads to

$$z_{k|ij}^{(s+1)} = \frac{P^{(s)}(t_j|\tau_k)P^{(s)}(\tau_k|d_i)}{\sum_{k'=1}^{L} P^{(s)}(t_j|\tau_{k'})P^{(s)}(\tau_{k'}|d_i)}, \tag{4.83}$$

where $P^{(s)}(t_j|\tau_k)$ and $P^{(s)}(\tau_k|d_i)$ are values of $P(t_j|\tau_k)$ and $P(\tau_k|d_i)$ in s-th step, respectively.

Updating formula for $P(\tau_k|d_i)$ in M-step can be obtained by solving $\partial\mathcal{H}/\partial P(\tau_k|d_i) = 0$ where $\mathcal{H} = \log L_c(\mathbf{\Psi}) - \lambda_k\{\sum_{k'} P(\tau_{k'}|d_i) - 1\}$ (λ_k is the Lagrange multiplier). Since

$$\frac{\partial\mathcal{H}}{\partial P(\tau_k|d_i)} = \sum_{j=1}^{M} \mathrm{f}_{ij} z_{k|ij} \frac{1}{P(t_j|\tau_k)P(\tau_k|d_i)} P(t_j|\tau_k) - \lambda_k, \tag{4.84}$$

from Equation (4.80), it follows that $P(\tau_k|d_i) = \lambda_k^{-1} \sum_j \mathrm{f}_{ij} z_{k|ij}$. Because λ_k is a normalizing factor, the final updating formula becomes

$$P^{(s)}(\tau_k|d_i) = \frac{\sum_{j=1}^{M} \mathrm{f}_{ij} z_{k|ij}^{(s)}}{\sum_{k'=1}^{L} \sum_{j=1}^{M} \mathrm{f}_{ij} z_{k'|ij}^{(s)}}. \tag{4.85}$$

Similarly, updating formula of $P(t_j|\tau_k)$ in M-step can be obtained such that

$$P^{(s)}(t_j|\tau_k) = \frac{\sum_{i=1}^{N} \mathrm{f}_{ij} z_{k|ij}^{(s)}}{\sum_{j'=1}^{M} \sum_{i=1}^{N} \mathrm{f}_{ij'} z_{k|ij'}^{(s)}}. \tag{4.86}$$

Procedure of DC based on the EM algorithm for PLSA is shown in Figure 4.9 as a summary. Actually, Table 4.13 indicates an experimental result when estimation of $P(t_j|\tau_k)$ and $P(\tau_k|d_i)$ on the sample DB was repeated 1000 times with changing initial values of these probabilities [39]. So many local maximums appeared, and the largest log likelihood score was -32.26, which is followed by -32.42, in this experiment. About 58% of runs reached to the two largest maximums, which would provide 'valid' clustering results, except that d_5 belongs to two clusters.

‡ Document clustering by PLSA

Set: The number of clusters L.

1) Initialize parameters $P(\tau_k|d_i)$ and $P(t_j|\tau_k)$ ($k = 1, \ldots, L$; $i = 1, \ldots, N$; $j = 1, \ldots, M$) by generating random numbers, and $s \leftarrow 0$.

2) [E-step] Compute $z_{k|ij}^{(s+1)}$ in Equation (4.83) ($k = 1, \ldots, L$; $i = 1, \ldots, N$; $j = 1, \ldots, M$), and $s \leftarrow s + 1$.

3) [M-step] Update $P(\tau_k|d_i)$ and $P(t_j|\tau_k)$ by Equations (4.85) and (4.86), respectively ($k = 1, \ldots, L$; $i = 1, \ldots, N$; $j = 1, \ldots, M$).

4) If parameters converge, then allocate each document to a cluster by Equation (4.77) and terminate the procedure. Otherwise, return to 2).

Out: Clusters $C_1, \ldots, C_L$ and estimated values of parameters.

Figure 4.9: Document clustering by PLSA

Examples of estimating $P(t_j|\tau_k)$ and $P(\tau_k|d_i)$ are shown in Tables 4.14 and 4.15, respectively, (where $\log L(\mathbf{\Psi}) = -32.26$). Since $P(\tau_1|d_5) = P(\tau_2|d_5) = 0.4$, document d_5 belongs to two clusters in Table 4.13. Documents d_6 and d_7 have also probabilities more than 0.0 for all three latent topics while $P(\tau_k|d_i)$ of the other documents concentrates on a single topic.

[39] Note that convergence of parameters computed by this algorithm seems to be slower than by the PMM and MMM (i.e., many more iterations were needed until convergence in this experiment) possibly because the number of parameters is larger.

Table 4.13: Experimental result of executing PLSA repeatedly ($L = 3$)

$\log L(\mathbf{\Psi})$	# of runs	clusters
-32.26	331	$\{d_1, d_2, d_3, d_5\}\{d_4, d_5, d_7, d_{10}\}\{d_6, d_8, d_9\}$
-32.42	247	$\{d_1, d_2, d_3, d_5\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$
-32.68	21	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7, d_{10}\}\{d_6, d_8, d_9\}$
-33.41	86	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$
-34.00	33	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7\}\{d_8, d_9, d_{10}\}$
-34.81	53	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7, d_9\}\{d_8, d_{10}\}$
-35.74	51	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7, d_8, d_9\}\{d_{10}\}$
-37.77	57	$\{d_1, d_2, d_3, d_4, d_5, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$
-38.09	16	$\{d_1, d_2, d_3, d_{10}\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$
-41.00	10	$\{d_1, d_{10}\}\{d_2, d_3, d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$
Others	95	-
Total	1000	

Note: 1. In total, 1000 runs were repeated using the sample DB.
2. "Others" includes 40 maximums appearing less than 10 runs.

Table 4.14: $P(t_j|\tau_k)$ when $\log L(\mathbf{\Psi}) = -32.26$

	t_1	t_2	t_3	t_4	t_5	t_6
τ_1	0.300	0.700	0.000	0.000	0.000	0.000
τ_2	0.000	0.000	0.800	0.200	0.000	0.000
τ_3	0.000	0.000	0.000	0.222	0.593	0.185

Table 4.15: $P(\tau_k|d_i)$ when $\log L(\mathbf{\Psi}) = -32.26$

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
τ_1	1.00	1.00	1.00	0.00	0.40	0.14	0.13	0.00	0.00	0.00
τ_2	0.00	0.00	0.00	1.00	0.40	0.14	0.63	0.00	0.00	0.00
τ_3	0.00	0.00	0.00	0.00	0.20	0.71	0.25	1.00	1.00	1.00

The PLSA may allow to categorize terms by using probability $P(t_j|\tau_k)$ shown in Table 4.14. For example, term t_j can be allocated to the k-th latent topic such that

$$k = \arg\max_{k'} P(t_j|\tau_{k'}), \tag{4.87}$$

which leads to a clustering result such as $\{t_1, t_2\}$, $\{t_3\}$ and $\{t_4, t_5, t_6\}$. Inevitably, in PLSA, the number of term clusters is always equal to that of document clusters (i.e., L is the number of latent topic classes in PLSA).

Extension of PLSA

As a natural extension of the PLSA, it may be possible to consider a model including two different types of the latent class for terms and documents, respectively. For instance, $P(t_j|d_i)$ can be assumed to be generated such that

$$P(t_j|d_i) = \sum_{k=1}^{L} \sum_{m=1}^{L'} P(t_j|\tilde{\tau}_m) P(\tilde{\tau}_m|\tau_k) P(\tau_k|d_i), \tag{4.88}$$

where $\tilde{\tau}_m$ and τ_k denote latent classes of terms and documents, respectively ($m = 1, \ldots, L'$; $k = 1, \ldots, L$).

Each probability can be formally obtained by an EM algorithm based on unobservable variable $z_{mk|ij}$ such that

$$z_{mk|ij} = \frac{P(t_j|\tilde{\tau}_m) P(\tilde{\tau}_m|\tau_k) P(\tau_k|d_i)}{\sum_{k'=1}^{L} \sum_{m'=1}^{L'} P(t_j|\tilde{\tau}_{m'}) P(\tilde{\tau}_{m'}|\tau_{k'}) P(\tau_{k'}|d_i)}, \tag{4.89}$$

which leads to

$$P(\tau_k|d_i) = \frac{\sum_{j=1}^{M} \sum_{m=1}^{L'} \mathrm{f}_{ij} z_{mk|ij}}{\sum_{k'=1}^{L} \sum_{j=1}^{M} \sum_{m=1}^{L'} \mathrm{f}_{ij} z_{mk'|ij}} \tag{4.90}$$

($P(t_j|\tilde{\tau}_m)$ and $P(\tilde{\tau}_m|\tau_k)$ can be computed in similar ways). However, it should be noted that there would be so many local maximums because $P(\tilde{\tau}_m|\tau_k)$ is completely free from observed data and determined within only the model [40].

In the sample DB, when $L' = 4$ and $L = 3$, the maximum log likelihood score of the model in Equation (4.88) would be also -32.26, which provides the same estimations on $P(\tau_k|d_i)$ with those in Table 4.15. Also, estimated values of $P(t_j|\tilde{\tau}_m)$ ($m = 1, 2, 3$) were also identical to those in Table 4.14. Meanwhile, values for an augmented latent topic $\tilde{\tau}_4$ were $P(t_1|\tilde{\tau}_4) = 0.000$, $P(t_2|\tilde{\tau}_4) = 0.138$, $P(t_3|\tilde{\tau}_4) = 0.177$, $P(t_4|\tilde{\tau}_4) = 0.511$, $P(t_5|\tilde{\tau}_4) = 0.174$ and $P(t_6|\tilde{\tau}_4) = 0.000$, which means that term clusters become $\{t_1, t_2\}$, $\{t_3\}$, $\{t_4\}$ and $\{t_5, t_6\}$. However, because it was estimated that $P(\tilde{\tau}_1|\tau_1) = 1.000$, $P(\tilde{\tau}_2|\tau_2) = 1.000$, $P(\tilde{\tau}_3|\tau_3) = 1.000$, and $P(\tilde{\tau}_4|\tau_k) = 0.000$ for $k = 1, 2, 3$, the forth term cluster has no contribution to computation of $P(t_j|d_i)$ in this case.

4.1.4 Latent Dirichlet allocation (LDA)

Generative model based on Dirichlet prior

PLSA tries to estimate each parameter $P(\tau_k|d_i)$ indicating a probability that document d_i is related with (or relevant to) latent topic τ_k (see Equation (4.76)), but it is possible to generate estimates over-fitted to observed data like in the case of $p_{j|k}$ for the standard (non-Bayesian) MMM. Actually, Table 4.13 shows that many local maximums were obtained by the EM algorithm for PLSA, which would be partly caused by calculating precisely values for all pairs of topics and documents.

[40] Another model has been proposed by [233] such that

$$P(t_j|d_i) = \sum_{k=1}^{L} \sum_{m=1}^{L'} P(t_j|\tilde{\tau}_m, \tau_k) P(\tilde{\tau}_m) P(\tau_k|d_i).$$

In the model, since $P(\tilde{\tau}_m)$ is perfectly free, different combinations of estimated values of $P(t_j|\tilde{\tau}_m, \tau_k)$ and $P(\tilde{\tau}_m)$ may provide the same likelihood score.

In particular, when each probability $P(t_j|\tau_k)$ is used for applications other than DC, the overfitting may become more serious problem. For instance, estimation of $P(t_j|\tau_k)$ allows to analyze a vocabulary in a corpus (e.g., a set of articles in scientific journals or of news papers), which helps us to understand a trend of topics to be focused on in the corpus. In such cases, $P(t_j|\tau_k)$ can be considered as a *topic model*. If the set of documents in a corpus is assumed as just a 'sample' for observing an underlying topic model, then it is possible that parameters estimated by using the set may be extraordinarily tailored to a particular sample.

One of the solutions is to assume that topic τ_k in a document is randomly chosen according to a common probability distribution, not to estimate directly $P(\tau_k|d_i)$ as a unique probability for a particular document. Then each term of the document is supposed to be generated based on the selected topic. This kind of generative probabilistic model was introduced by Blei, Ng & Jordan(2003) [30], which is called *latent Dirichlet allocation* (LDA) and has been widely used in various areas (e.g., DC, NLP, data mining, and so on).

In the LDA, the generation of a document is formalized on a sequence of word tokens w_{ih} in d_i ($h = 1, \ldots, l_i$ and l_i means document length) such that

$$\mathbf{w}_i = [\mathrm{w}_{i1}, \ldots, \mathrm{w}_{il_i}]^T = [t_{\omega(1|i)}, \ldots, t_{\omega(l_i|i)}]^T, \tag{4.91}$$

where $\omega(h|i)$ indicates index number of a term appearing in h-th position in document d_i. For example, if $\omega(4|i) = 8$, then the 4th word token in document d_i is t_8. Similarly, index number of a latent topic for token w_{ih} is denoted by $\tilde{z}_{ih}$ here (e.g., if 4th token in d_i is generated from τ_2, then $\tilde{z}_{i4} = 2$), which means that a latent topic is assigned to each token, not each term [41].

Each token in d_i is assumed to be randomly selected from a discrete distribution,

$$\boldsymbol{\phi}_k \equiv [\phi_{1|k}, \ldots, \phi_{M|k}]^T = [P(t_1|\tau_k), \ldots, P(t_M|\tau_k)]^T, \tag{4.92}$$

after determining randomly τ_k as a topic of the token based on another discrete distribution,

$$\boldsymbol{\theta}_i = [\theta_{i1}, \ldots, \theta_{iL}]^T, \tag{4.93}$$

element θ_{ik} of which indicates a probability that k-th topic is chosen for d_i (i.e., $\sum_k \theta_{ik} = 1$). For example, if first token of d_4 is t_3, then it can be imagined that

$$d_4 \to \boldsymbol{\theta}_4 \to \pi(\cdot|\boldsymbol{\theta}_4) \mapsto \tau_2 \to \boldsymbol{\phi}_2 \to \pi(\cdot|\boldsymbol{\phi}_2) \mapsto t_3 \to \mathrm{w}_{41},$$

in which τ_2 and t_3 are assumed to be drawn from distributions $\boldsymbol{\theta}_4$ and $\boldsymbol{\phi}_2$, respectively (e.g., $\pi(\cdot|\boldsymbol{\theta}_4)$ represents a discrete distribution, $P(1) = \theta_{41}, \ldots, P(L) = \theta_{4L}$).

Furthermore, in the LDA model, $\boldsymbol{\phi}_k$ and $\boldsymbol{\theta}_i$ are supposed to be drawn from Dirichlet distributions, and therefore, a process of generating d_i becomes as follows [30, 109, 29]:

1. Choose $\boldsymbol{\phi}_k$ from a Dirichlet distribution, $\boldsymbol{\phi}_k \sim P_D(\cdot|\boldsymbol{\beta})$.
2. Choose $\boldsymbol{\theta}_i$ from a Dirichlet distribution, $\boldsymbol{\theta}_i \sim P_D(\cdot|\boldsymbol{\alpha})$.
3. For each token w_{ih} ($h = 1, \ldots, l_i$),
   (a) Choose a topic $\tilde{z}_{ih}$ from a discrete distribution, $\tilde{z}_{ih} \sim \pi(\cdot|\boldsymbol{\theta}_i)$,
   (b) Choose a term from $\{t_1, \ldots, t_M\}$ as token w_{ih} according to a discrete distribution conditioned on the topic $\tilde{z}_{ih}$, $\mathrm{w}_{ih} \sim \pi(\cdot|\tilde{z}_{ih}, \boldsymbol{\phi})$.

Particularly, P_D denotes a Dirichlet distribution here, and $\boldsymbol{\phi}$ is defined by

$$\boldsymbol{\phi} = [\boldsymbol{\phi}_1^T, \ldots, \boldsymbol{\phi}_L^T]^T = [P(t_1|\tau_1), \ldots, P(t_M|\tau_L)]^T. \tag{4.94}$$

As described above, notation $\tilde{z}_{ih} \sim \pi(\cdot|\boldsymbol{\theta}_i)$ means that a topic number k is randomly selected from a discrete distribution $\pi(k) = \theta_{ik}$ ($k = 1, \ldots, L$) for d_i, and $\mathrm{w}_{ih} \sim \pi(\cdot|\tilde{z}_{ih}, \boldsymbol{\phi})$ implies that the discrete distribution for selecting a term is actually $\pi(j) = P(t_j|\tau_k)$ ($j = 1, \ldots, M$) under a result that $\tilde{z}_{ih} = k$ in step 3(a). Figure 4.10 shows graphical models describing the PLSA and LDA, respectively (note that l in the figure denotes 'collection length', which is defined by $l = \sum_i l_i$).

[41] Therefore, when a term appears twice in a single document, it may be possible that their latent topics are different.

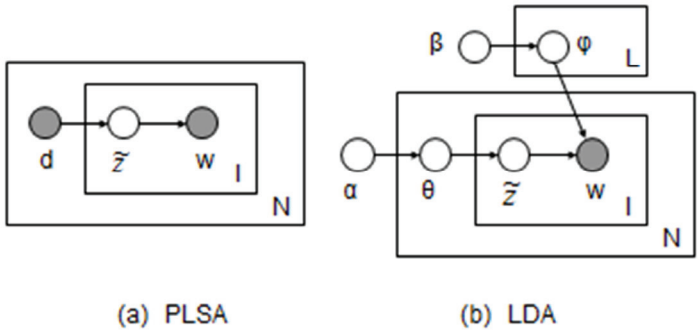


Figure 4.10: Graphical model representation of PLSA and LDA

In the framework of LDA, the basic formula of PLSA in Equation (4.76) changes to

$$P(\mathrm{w}_{ih}|\boldsymbol{\phi},\boldsymbol{\theta}_i) = \sum_{k=1}^{L} P(\mathrm{w}_{ih}|\tilde{z}_{ih}=k,\boldsymbol{\phi})P(\tilde{z}_{ih}=k|\boldsymbol{\theta}_i), \tag{4.95}$$

under an assumption that $\tilde{z}_{ih}$ is independent of $\boldsymbol{\phi}$. Therefore, when it is simply assumed that $P(\mathbf{w}_i) = \prod_h P(\mathrm{w}_{ih})$, the posterior distribution of LDA model can be written as

$$P(\mathbf{w}_i|\boldsymbol{\alpha},\boldsymbol{\beta}) = \int\int P(\boldsymbol{\phi}|\boldsymbol{\beta})P(\boldsymbol{\theta}_i|\boldsymbol{\alpha}) \left[\prod_{h=1}^{l_i}\sum_{\tilde{z}_{ih}=1}^{L} P(\mathrm{w}_{ih}|\tilde{z}_{ih},\boldsymbol{\phi})P(\tilde{z}_{ih}|\boldsymbol{\theta}_i)\right] d\boldsymbol{\theta}_i d\boldsymbol{\phi}, \tag{4.96}$$

by integrating out $\boldsymbol{\phi}$ and $\boldsymbol{\theta}_i$ $(i = 1,\ldots,N)$ [30]. Also, if observed data of tokens in D is denoted by $\mathbf{w} = [\mathbf{w}_1^T,\ldots,\mathbf{w}_N^T]^T$, then a probabilistic model of D can be represented as

$$P(\mathbf{w}|\boldsymbol{\alpha},\boldsymbol{\beta}) = \prod_{i=1}^{N} P(\mathbf{w}_i|\boldsymbol{\alpha},\boldsymbol{\beta}), \tag{4.97}$$

by using Equation (4.96) under an assumption of document independence.

Estimation by using variational parameters

Whereas it is indispensable to estimate $\boldsymbol{\phi}$ in exploring a topic model, estimation of $\boldsymbol{\theta}_i$ $(i = 1,\ldots,N)$ is more important from a viewpoint of DC because a cluster set can be naturally generated as

$$C_k = \left\{ d_i \,\middle|\, k = \underset{k'=1,\ldots,L}{\arg\max}\, \hat{\theta}_{ik'} \right\}, \tag{4.98}$$

if estimates $\hat{\theta}_{ik}$ $(i = 1,\ldots,N;\ k = 1,\ldots,L)$ were obtained from the observed data. However, the posterior distribution in Equation (4.96) is intractable, and therefore, it is hard to compute the MLEs based on Equations (4.96) and (4.97).

One of the solutions is to employ a *variational distribution* q, which allows to have a lower bound of the log likelihood computed from Equation (4.96) (i.e., $\log P(\mathbf{w}_i|\boldsymbol{\alpha},\boldsymbol{\beta})$). Blei, Ng & Jordan (2003) [30] provided a detailed explanation on a technique of parameter estimation based on the variational distribution. If topic assignment to each token in d_i is denoted by $\tilde{\mathbf{z}}_i = [\tilde{z}_{i1},\ldots,\tilde{z}_{il_i}]^T$, then the logarithm of Equation (4.96) becomes that

$$\log P(\mathbf{w}_i|\boldsymbol{\alpha},\boldsymbol{\beta}) = \log \int\int \sum_{\tilde{\mathbf{z}}_i} P(\mathbf{w}_i,\boldsymbol{\theta}_i,\boldsymbol{\phi},\tilde{\mathbf{z}}_i|\boldsymbol{\alpha},\boldsymbol{\beta}) d\boldsymbol{\theta}_i d\boldsymbol{\phi}, \tag{4.99}$$

where $\sum_{\tilde{\mathbf{z}}_i} = \sum_{\tilde{z}_{i1}} \cdots \sum_{\tilde{z}_{il_i}}$ [42]. Similarly, since all documents in a set are assumed to be independent each other, index i can be simply removed from the equation for the entire set such that

$$\log P(\mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \log \int \int \sum_{\tilde{\mathbf{z}}} P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\alpha}, \boldsymbol{\beta}) d\boldsymbol{\theta} d\boldsymbol{\phi}, \tag{4.100}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_N^T]^T$ and $\tilde{\mathbf{z}} = [\tilde{\mathbf{z}}_1^T, \ldots, \tilde{\mathbf{z}}_N^T]^T$.

A strategy of the technique for estimation of parameters is to insert a variational distribution $q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})$ into the equation such that

$$\log P(\mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \log \int \int \sum_{\tilde{\mathbf{z}}} \frac{P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\alpha}, \boldsymbol{\beta}) q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})}{q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})} d\boldsymbol{\theta} d\boldsymbol{\phi}, \tag{4.101}$$

which leads to an inequality

$$\log P(\mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \geq E_q[\log P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\alpha}, \boldsymbol{\beta})] - E_q[\log q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})] \equiv \mathcal{L}_q \tag{4.102}$$

where E_q means an expectation by q [43]. For instance, if it is assumed that x and z are continuous and discrete variables, respectively, then the expectation of a joint probability distribution $P(x, z)$ by $q(x, z)$ is computed as

$$E_q[P(x, z)] = \int \sum_z P(x, z) q(x, z) dx. \tag{4.103}$$

Since $\mathcal{L}_q$ in Equation (4.102) gives a lower bound of $\log P(\mathbf{w}|\boldsymbol{\alpha}, \boldsymbol{\beta})$, it is possible to estimate parameters in LDA model by maximizing $\mathcal{L}_q$ instead of intractable Equations (4.96) and (4.97).

Needless to say, for proper estimation, it is necessary to select an appropriate distribution as q. Blei, Ng & Jordan(2003) [30] proposed to use

$$q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\gamma}, \boldsymbol{\varphi}, \boldsymbol{\nu}) = \prod_{k=1}^{L} q(\boldsymbol{\phi}_k|\boldsymbol{\nu}_k) \times \prod_{i=1}^{N} q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i) \prod_{h=1}^{l_i} q(\tilde{z}_{ih}|\boldsymbol{\varphi}_{ih}), \tag{4.104}$$

where $\boldsymbol{\nu}$, $\boldsymbol{\gamma}$, and $\boldsymbol{\varphi}$ are called *variational parameters*. More specifically, the parameters are defined such that $\boldsymbol{\nu}_k = [\nu_{1k}, \ldots, \nu_{Mk}]^T$ and $\boldsymbol{\nu} = [\boldsymbol{\nu}_1^T, \ldots, \boldsymbol{\nu}_L^T]^T$, and $\boldsymbol{\gamma}_i = [\gamma_{i1}, \ldots, \gamma_{iL}]^T$ and $\boldsymbol{\gamma} = [\boldsymbol{\gamma}_1^T, \ldots, \boldsymbol{\gamma}_N^T]^T$. Also, $\boldsymbol{\varphi}_{ih} = [\varphi_{ih1}, \ldots, \varphi_{ihL}]^T$ and $\boldsymbol{\varphi} = [\boldsymbol{\varphi}_{11}^T, \ldots, \boldsymbol{\varphi}_{Nl_N}^T]^T$. In Equation (4.104), $q(\boldsymbol{\phi}_k|\boldsymbol{\nu}_k)$ and $q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i)$ are assumed to be Dirichlet distributions, and $q(\cdot|\boldsymbol{\varphi}_{ih})$ is a discrete distribution.

By factorizing P and q with considering dependency between parameters,

$$\begin{aligned} \mathcal{L}_q &= E_q[\log P(\boldsymbol{\theta}|\boldsymbol{\alpha})] + E_q[\log P(\tilde{\mathbf{z}}|\boldsymbol{\theta})] + E_q[\log P(\mathbf{w}|\tilde{\mathbf{z}}, \boldsymbol{\phi})] + E_q[\log P(\boldsymbol{\phi}|\boldsymbol{\beta})] \\ &\quad - E_q[\log q(\boldsymbol{\theta})] - E_q[\log q(\tilde{\mathbf{z}})] - E_q[\log q(\boldsymbol{\phi})], \end{aligned} \tag{4.105}$$

is obtained [44]. Each term $E_q[\cdot]$ in the equation can be calculated as follows. First,

$$E_q[\log P(\boldsymbol{\theta}|\boldsymbol{\alpha})] = \sum_{i=1}^{N} \int [\log P(\boldsymbol{\theta}_i|\boldsymbol{\alpha})] \, q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i) d\boldsymbol{\theta}_i = \sum_{i=1}^{N} E_q[\log P(\boldsymbol{\theta}_i|\boldsymbol{\alpha})|\boldsymbol{\gamma}_i], \tag{4.106}$$

[42] If $l_i = 2$, then $\sum_{\tilde{z}_1} \sum_{\tilde{z}_2} P(\mathrm{w}_1, \tilde{z}_1) P(\mathrm{w}_2, \tilde{z}_2) = \sum_{\tilde{z}_1} P(\mathrm{w}_1, \tilde{z}_1) \sum_{\tilde{z}_2} P(\mathrm{w}_2, \tilde{z}_2)$ due to the assumption of independence between tokens (note that index i is omitted here for simplicity). This implies that $\sum_{\tilde{\mathbf{z}}} \prod_h P(\mathrm{w}_h, \tilde{z}_h) = \prod_h \sum_{\tilde{z}_h} P(\mathrm{w}_h, \tilde{z}_h)$.

[43] For simplicity, hyperparameters are omitted here. If an expectation is defined by Equation (4.103), then by using Jensen's inequality, Equation (4.102) can be written as

$$\log \int \int \sum_{\tilde{\mathbf{z}}} \frac{P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})}{q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})} q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}) d\boldsymbol{\theta} d\boldsymbol{\phi} \geq \int \int \sum_{\tilde{\mathbf{z}}} q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}) \log \frac{P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})}{q(\boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}})} d\boldsymbol{\theta} d\boldsymbol{\phi},$$

and the right-hand side of this equation means $\mathcal{L}_q$ in Equation (4.102). Note that $g[E(x)] \geq E[g(x)]$ if $E(x)$ is finite and g is a concave function, which is called Jensen's inequality. In this case, $g(\cdot) = \log$, and the logarithm is a concave function.

[44] In the factorization, it is assumed that

$$P(\mathbf{w}, \boldsymbol{\theta}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = P(\mathbf{w}, \boldsymbol{\phi}, \tilde{\mathbf{z}}|\boldsymbol{\theta}, \boldsymbol{\beta}) P(\boldsymbol{\theta}|\boldsymbol{\alpha}) = P(\mathbf{w}, \tilde{\mathbf{z}}|\boldsymbol{\phi}, \boldsymbol{\theta}) P(\boldsymbol{\phi}|\boldsymbol{\beta}) P(\boldsymbol{\theta}|\boldsymbol{\alpha}) = P(\mathbf{w}|\tilde{\mathbf{z}}, \boldsymbol{\phi}) P(\tilde{\mathbf{z}}|\boldsymbol{\theta}) P(\boldsymbol{\phi}|\boldsymbol{\beta}) P(\boldsymbol{\theta}|\boldsymbol{\alpha}),$$

according to the dependency. Also, note that $E[\log(xy)] = E(\log x) + E(\log y)$ by the definition of expectation.

where

$$\log P(\boldsymbol{\theta}_i|\boldsymbol{\alpha}) = \log\Gamma\left(\sum_{k=1}^{L}\alpha_k\right) - \sum_{k=1}^{L}\log\Gamma(\alpha_k) + \sum_{k=1}^{L}(\alpha_k - 1)\log\theta_{ik}, \tag{4.107}$$

because $P(\boldsymbol{\theta}_i|\boldsymbol{\alpha})$ is a Dirichlet distribution. When $q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i)$ is a Dirichlet distribution,

$$E_q[\log\theta_{ik}|\boldsymbol{\gamma}_i] = \int \log\theta_{ik} q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i) d\boldsymbol{\theta}_i = \psi(\gamma_{ik}) - \psi\left(\sum_{k'=1}^{L}\gamma_{ik'}\right), \tag{4.108}$$

where $\psi(x) = d\log\Gamma(x)/dx$ which is generally called digamma function [45]. Thus Equation (4.106) becomes that

$$\begin{aligned} E_q[\log P(\boldsymbol{\theta}|\boldsymbol{\alpha})] &= \sum_{i=1}^{N}\left[\log\Gamma\left(\sum_{k=1}^{L}\alpha_k\right) - \sum_{k=1}^{L}\log\Gamma(\alpha_k)\right. \\ &\quad \left. + \sum_{k=1}^{L}(\alpha_k - 1)\left[\psi(\gamma_{ik}) - \psi\left(\sum_{k'=1}^{L}\gamma_{ik'}\right)\right]\right]. \end{aligned} \tag{4.109}$$

Second,

$$E_q[\log P(\tilde{\mathbf{z}}|\boldsymbol{\theta})] = \sum_{i=1}^{N}\sum_{h=1}^{l_i} E_q[\log P(\tilde{z}_{ih}|\boldsymbol{\theta}_i)], \tag{4.110}$$

where

$$E_q[\log P(\tilde{z}_{ih}|\boldsymbol{\theta}_i)] = \sum_{k=1}^{L}\varphi_{ihk}\left[\psi(\gamma_{ik}) - \psi\left(\sum_{k'=1}^{L}\gamma_{ik'}\right)\right], \tag{4.111}$$

because $q(\cdot|\boldsymbol{\varphi}_i)$ is a discrete distribution [46].

Third,

$$E_q[\log P(\mathbf{w}|\tilde{\mathbf{z}}, \boldsymbol{\phi})] = \sum_{i=1}^{N}\sum_{h=1}^{l_i} E_q[\log P(\mathrm{w}_{ih}|\tilde{z}_{ih}, \boldsymbol{\phi})], \tag{4.112}$$

where

$$E_q[\log P(\mathrm{w}_{ih} = t_j|\tilde{z}_{ih}, \boldsymbol{\phi})] = \sum_{k=1}^{L}\varphi_{ihk}\left[\psi(\nu_{jk}) - \psi\left(\sum_{j'=1}^{M}\nu_{j'k}\right)\right], \tag{4.113}$$

for each token [47].

---

[45] This expectation can be derived from the natural parameterization of the exponential family representation of the Dirichlet distribution (see [30] for details).

[46] From the definition,

$$\begin{aligned} E_q[\log P(\tilde{z}_{ih}|\boldsymbol{\theta}_i)] &= \int\sum_{k=1}^{L}\log P(\tilde{z}_{ih} = k|\boldsymbol{\theta}_i) q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i) q(\tilde{z}_{ih} = k|\boldsymbol{\varphi}_i) d\boldsymbol{\theta}_i \\ &= \sum_{k=1}^{L}\varphi_{ihk}\int\log\theta_{ik} q(\boldsymbol{\theta}_i|\boldsymbol{\gamma}_i) d\boldsymbol{\theta}_i. \end{aligned}$$

By using Equation (4.108), Equation (4.111) can be obtained.

[47] When t_j is used as token w_{ih}, it becomes

$$\begin{aligned} E_q[\log P(\mathrm{w}_{ih} = t_j|\tilde{z}_{ih}, \boldsymbol{\phi})] &= \int\sum_{k=1}^{L}\log P(\mathrm{w}_{ih} = t_j|\tilde{z}_{ih} = k, \boldsymbol{\phi}_k) q(\boldsymbol{\phi}_k|\boldsymbol{\nu}_k) q(\tilde{z}_{ih} = k|\boldsymbol{\varphi}_i) d\boldsymbol{\phi}_k \\ &= \sum_{k=1}^{L}\varphi_{ihk}\int\log\phi_{jk} q(\boldsymbol{\phi}_k|\boldsymbol{\nu}_k) d\boldsymbol{\phi}_k. \end{aligned}$$

According to Equation (4.108), Equation (4.113) is obtained.

Forth, $E_q[\log P(\boldsymbol{\phi}|\boldsymbol{\beta})]$ can be computed in a similar way to calculation of $E_q[\log P(\boldsymbol{\theta}|\boldsymbol{\alpha})]$, which consequently leads to

$$\begin{aligned} E_q[\log P(\boldsymbol{\phi}|\boldsymbol{\beta})] &= \sum_{k=1}^{L}\left[\log\Gamma\left(\sum_{j=1}^{M}\beta_j\right) - \sum_{j=1}^{M}\log\Gamma(\beta_j) \right. \\ &\quad \left. + \sum_{j=1}^{M}(\beta_j - 1)\left[\psi(\nu_{jk}) - \psi\left(\sum_{j'=1}^{M}\nu_{j'k}\right)\right]\right]. \end{aligned} \tag{4.114}$$

Finally, it is necessary to compute $E_q[\log q(\boldsymbol{\theta})]$, $E_q[\log q(\tilde{\mathbf{z}})]$ and $E_q[\log q(\boldsymbol{\phi})]$ in Equation (4.105), which are expectations of q itself. Among them, $E_q[\log q(\boldsymbol{\theta})]$ and $E_q[\log q(\boldsymbol{\phi})]$ are again computed in a similar way to $E_q[\log P(\boldsymbol{\theta}|\boldsymbol{\alpha})]$, namely,

$$\begin{aligned} E_q[\log q(\boldsymbol{\theta})] &= \sum_{i=1}^{N}\left[\log\Gamma\left(\sum_{k=1}^{L}\gamma_{ik}\right) - \sum_{k=1}^{L}\log\Gamma(\gamma_{ik}) \right. \\ &\quad \left. + \sum_{k=1}^{L}(\gamma_{ik} - 1)\left[\psi(\gamma_{ik}) - \psi\left(\sum_{k'=1}^{L}\gamma_{ik'}\right)\right]\right], \end{aligned} \tag{4.115}$$

and

$$\begin{aligned} E_q[\log q(\boldsymbol{\phi})] &= \sum_{k=1}^{L}\left[\log\Gamma\left(\sum_{j=1}^{M}\nu_{jk}\right) - \sum_{j=1}^{M}\log\Gamma(\nu_{jk}) \right. \\ &\quad \left. + \sum_{j=1}^{M}(\nu_{jk} - 1)\left[\psi(\nu_{jk}) - \psi\left(\sum_{j'=1}^{M}\nu_{j'k}\right)\right]\right], \end{aligned} \tag{4.116}$$

because $q(\boldsymbol{\theta})$ and $q(\boldsymbol{\phi})$ are also Dirichlet distributions. In contrast, since $q(\tilde{\mathbf{z}})$ is a discrete distribution,

$$E_q[\log q(\tilde{\mathbf{z}})] = \sum_{i=1}^{N}\sum_{h=1}^{l_i}\sum_{k=1}^{L}\varphi_{ihk}\log\varphi_{ihk}, \tag{4.117}$$

is obtained.

The MLEs of parameters in the LDA model can be approximately obtained by maximizing the lower bound in Equation (4.105) for each parameter, which means that its derivatives have to be calculated. For instance, a variational parameter γ_{ik} appears in Equations (4.109), (4.111) and (4.115), and it follows that

$$\frac{\partial\mathcal{L}_q}{\partial\gamma_{ik}} = \psi'(\gamma_{ik})\left(\alpha_k + \sum_{h=1}^{l_i}\varphi_{ihk} - \gamma_{ik}\right) - \psi'\left(\sum_{k'=1}^{L}\gamma_{ik'}\right)\sum_{k'=1}^{L}\left(\alpha_{k'} + \sum_{h=1}^{l_i}\varphi_{ihk'} - \gamma_{ik'}\right), \tag{4.118}$$

where $\psi'(x) = d\psi(x)/dx$. The derivative $\partial\mathcal{L}_q/\partial\gamma_{ik}$ is always zero if

$$\gamma_{ik} = \alpha_k + \sum_{h=1}^{l_i}\varphi_{ihk}, \tag{4.119}$$

for $k = 1, \ldots, L$, which can be considered as an estimate of γ_{ik} [30]. Also, from Equations (4.111), (4.113) and (4.117), the derivative with respect to φ_{hk} for a token such that $\mathrm{w}_{ih} = t_j$ becomes that

$$\begin{aligned} \frac{\partial\mathcal{L}_q}{\partial\varphi_{ihk}} &= \left[\psi(\gamma_{ik}) - \psi\left(\sum_{k'=1}^{L}\gamma_{ik'}\right)\right] + \left[\psi(\nu_{jk}) - \psi\left(\sum_{j'=1}^{M}\nu_{j'k}\right)\right] \\ &\quad - \log\varphi_{ihk} - 1 + \lambda, \end{aligned} \tag{4.120}$$

where λ is the Lagrange multiplier for a condition that $\sum_k \varphi_{ihk} = 1$. Therefore, φ_{ihk} can be estimated such that $\varphi_{ihk} = \tilde{\varphi}_{ihk} / \sum_{k'} \tilde{\varphi}_{ihk'}$ where

$$\tilde{\varphi}_{ihk} = \exp\left[\psi(\gamma_{ik}) + \psi(\nu_{jk}) - \psi\left(\sum_{j'=1}^{M} \nu_{j'k}\right)\right], \tag{4.121}$$

for a particular token w_{ih} (note that $\psi(\sum_k \gamma_{ik})$ is canceled out in the normalization of $\tilde{\varphi}_{ihk}$).

Finally, by comparing Equation (4.111) with (4.113), (4.109) with (4.114) and (4.115) with (4.116), it is clear that the derivative of $\mathcal{L}_q$ with respect to ν_{jk} takes the almost similar form to that with respect to γ_{ik}. Thus from $\partial \mathcal{L}_q / \partial \nu_{jk} = 0$,

$$\nu_{jk} = \beta_k + \sum_{i=1}^{N} \sum_{h=1}^{l_i} I(\mathrm{w}_{ih} = t_j) \varphi_{ihk}, \tag{4.122}$$

is obtained where $I(\mathrm{w}_{ih} = t_j)$ is an indicator function returning 1 if $\mathrm{w}_{ih} = t_j$, and otherwise 0 [29].

By computing iteratively Equations (4.119), (4.121) and (4.122) for a given dataset until they converge, estimation of variational parameters becomes feasible. After that, θ_{ik} in Equation (4.98) can be estimated as

$$\hat{\theta}_{ik} = \frac{\gamma_{ik}}{\sum_{k'=1}^{L} \gamma_{ik'}}, \tag{4.123}$$

and similarly, $\hat{\phi}_{j|k} = \hat{P}(t_j|\tau_k) = \nu_{jk} / \sum_{j'=1}^{M} \nu_{j'k}$ [29]. Figure 4.11 is a summary of the DC algorithm based on variational parameters for LDA model.

‡ Document clustering based on variational parameters for LDA model

Set: The number of clusters L, and hyperparameter vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

1) Initialize φ_{ihk} for all tokens, all topics (clusters) and for all documents.

2) Update ν_{jk} based on Equation (4.122) for $j = 1, \ldots, M$ and $k = 1, \ldots, L$.

3) For each document ($i = 1, \ldots, N$),

   3-1) Update γ_{ik} using Equation (4.119) for $k = 1, \ldots, L$.

   3-2) Update φ_{ihk} such that $\varphi_{ihk} = \tilde{\varphi}_{ihk} / \sum_{k'} \tilde{\varphi}_{ihk'}$ where $\tilde{\varphi}_{ihk}$ is computed by Equation (4.121) ($h = 1, \ldots, l_i$; $k = 1, \ldots, L$).

4) If variational parameters converge, then generate clusters based on Equations (4.123) and (4.98), and terminate the procedure. Otherwise, return to step 2).

Out: Clusters $C_1, \ldots, C_L$ and estimates of $\phi_{j|k}$ and θ_{ik}.

Figure 4.11: Document clustering based on variational parameters for LDA model

Table 4.16 shows an example of estimating variational parameters for the LDA model by applying the algorithm in Figure 4.11 to the sample DB, in which $\alpha_k = 0.1$ and $\beta_j = 0.1$ were fixed for all k and j (i.e., all Dirichlet distributions are symmetric in this experiment) [48]. It seems that a valid set of clusters was obtained by the algorithm. However it should be noted that there would be many local maximums in the variational inference of the LDA model partly because many variational parameters are included in the computation. For example, Figure 4.12 shows a result of an experiment in which clustering by variational parameters for the LDA model was independently

[48] A technique of estimating empirically hyperparameters from the data using the Newton-Raphson method was also provided by [30]. Since actual values of the hyperparameters have a strong influence on its result (i.e., results may be largely changed by using another set of values), it should be carefully to set actual values for the hyperparameters if they are not empirically estimated. This problem will be discussed in the section of estimation by Gibbs sampling.

Table 4.16: Examples of estimating LDA model by variational parameters

	$\hat{\theta}_{ik}$				$\hat{\phi}_{j\|k}$		
	C_1	C_2	C_3		C_1	C_2	C_3
d_1	.962	.019	.019	t_1	.318	.005	.004
d_2	.953	.023	.023	t_2	.661	.076	.004
d_3	.973	.014	.014	t_3	.005	.599	.004
d_4	.014	.973	.014	t_4	.005	.153	.250
d_5	.337	.644	.019	t_5	.005	.112	.570
d_6	.133	.191	.676	t_6	.005	.054	.168
d_7	.012	.976	.012				
d_8	.019	.019	.962				
d_9	.016	.016	.968				
d_{10}	.001	.001	.976				

Note: 1. $\alpha_k = 0.1$ and $\beta_j = 0.1$ for all k and j.
2. The number of iterations was 29.

repeated 1000 times for the sample DB. It turns out that many local maximums appeared and that only 28.8% of 1000 runs reached to 'valid' cluster set, $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7, d_{10}\}, \{d_6, d_8, d_9\}\}$ or $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$, in this experiment.

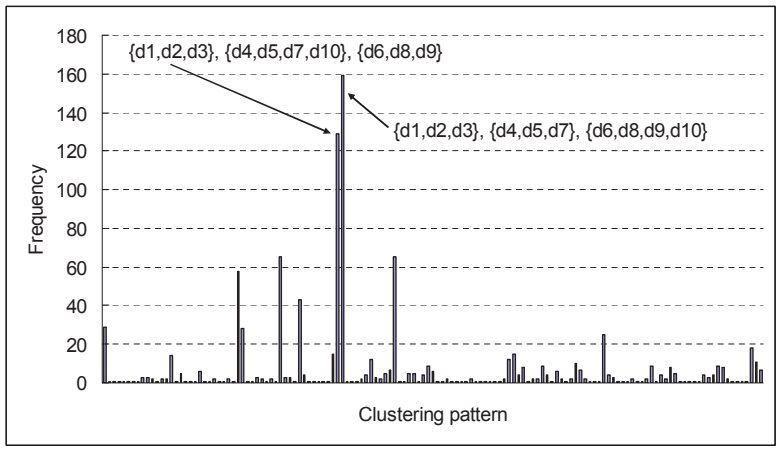


Figure 4.12: Clustering results by estimated variational parameters of LDA (1000 runs)

Estimation by Gibbs sampling

Another method for inferencing parameters of LDA model was given by Griffiths & Steyvers(2004) [109], which has been widely used by other researchers. Its strategy is to use a Gibbs sampling method for estimating distribution $P(\tilde{\mathbf{z}}|\mathbf{w}) = \prod_i P(\tilde{\mathbf{z}}_i|\mathbf{w}_i)$ where $\tilde{\mathbf{z}}_i = [\tilde{z}_{i1}, \ldots, \tilde{z}_{il_i}]^T$ and $\tilde{\mathbf{z}} = [\tilde{\mathbf{z}}_1^T, \ldots, \tilde{\mathbf{z}}_N^T]^T$, each term of which can be computed as

$$P(\tilde{\mathbf{z}}_i|\mathbf{w}_i) = \frac{P(\tilde{\mathbf{z}}_i, \mathbf{w}_i)}{P(\mathbf{w}_i)} = \frac{P(\mathbf{w}_i|\tilde{\mathbf{z}}_i)P(\tilde{\mathbf{z}}_i)}{P(\mathbf{w}_i)}. \tag{4.124}$$

As described below, although parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}_i$ are integrated out in process of calculating analytically full conditional distributions for sampling, it is possible to estimate them empirically from samples obtained through iterative steps of Gibbs sampling for computation of $P(\tilde{\mathbf{z}}|\mathbf{w})$.

When using explicitly the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}_i$, two probabilities in the numerator of the most right-hand side in Equation (4.124) can be written as

$$P(\mathbf{w}_i|\tilde{\mathbf{z}}_i, \boldsymbol{\phi}) = \prod_{h=1}^{l_i} \phi[\omega(h|i)|\tilde{z}_{ih}], \tag{4.125}$$

where $\phi[j|k] \equiv \phi_{j|k} = P(t_j|\tau_k)$, and

$$P(\tilde{\mathbf{z}}_i|\boldsymbol{\theta}_i) = \prod_{h=1}^{l_i} \theta[\tilde{z}_{ih}], \tag{4.126}$$

where $\theta[k] \equiv \theta_k$, respectively, according to LDA model. If a symmetric Dirichlet distribution such that

$$P(\boldsymbol{\phi}_k|\beta) = \frac{\Gamma(\beta M)}{\Gamma(\beta)^M} \prod_{j=1}^{M} \phi_{j|k}^{\beta-1} \tag{4.127}$$

(i.e., $\boldsymbol{\beta} = [\beta, \ldots, \beta]^T$) is assumed as $P_D(\cdot|\boldsymbol{\beta})$ for $\boldsymbol{\phi}_k$, then since $P(\boldsymbol{\phi}) = P(\boldsymbol{\phi}_1) \times \ldots \times P(\boldsymbol{\phi}_L)$, it follows that

$$P(\mathbf{w}_i|\tilde{\mathbf{z}}_i) = \int P(\mathbf{w}_i|\tilde{\mathbf{z}}_i, \boldsymbol{\phi}) P(\boldsymbol{\phi}) d\boldsymbol{\phi} \propto \prod_{k=1}^{L} \int \prod_{h=1}^{l_i} \phi[\omega(h|i)|\tilde{z}_{ih}] \prod_{j=1}^{M} \phi_{j|k}^{\beta-1} d\boldsymbol{\phi}_k = \prod_{k=1}^{L} \frac{\prod_j \Gamma(\mathrm{f}_{j|ik} + \beta)}{\Gamma(\mathrm{f}_{\cdot|ik} + M\beta)}, \tag{4.128}$$

where $\mathrm{f}_{j|ik}$ indicates the times that t_j appears as a token of document d_i when $\tilde{z}_{ih} = k$ ($h = 1, \ldots, l_i$), and $\mathrm{f}_{\cdot|ik} = \sum_j \mathrm{f}_{j|ik}$ [109] [49]. Similarly, when the Dirichlet distribution $P(\boldsymbol{\theta}_i|\boldsymbol{\alpha})$ is also symmetric (i.e., $\boldsymbol{\alpha} = [\alpha, \ldots, \alpha]^T$),

$$P(\tilde{\mathbf{z}}_i) = \int P(\tilde{\mathbf{z}}_i|\boldsymbol{\theta}_i) P(\boldsymbol{\theta}_i)\, d\boldsymbol{\theta}_i \propto \frac{\prod_k \Gamma(\mathrm{f}_{k|i} + \alpha)}{\Gamma(\mathrm{f}_{\cdot|i} + L\alpha)}, \tag{4.129}$$

is also obtained where $\mathrm{f}_{k|i}$ denotes the number of tokens belonging to topic k in d_i, and $\mathrm{f}_{\cdot|i} = \sum_k \mathrm{f}_{k|i} = l_i$.

The samples for estimating distribution $P(\tilde{\mathbf{z}}|\mathbf{w})$ can be drawn from full conditional probability,

$$P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}^{\neg ih}, \mathbf{w}) = \frac{P(\tilde{z}_{ih} = k, \tilde{\mathbf{z}}^{\neg ih}|\mathbf{w})}{P(\tilde{\mathbf{z}}^{\neg ih}|\mathbf{w})},\ k = 1, \ldots, L, \tag{4.130}$$

where $\tilde{\mathbf{z}}^{\neg ih}$ denotes a vector created by removing particular single element $\tilde{z}_{ih}$ from $\tilde{\mathbf{z}}$. Since $P(\tilde{\mathbf{z}}|\mathbf{w}) = \prod_i P(\tilde{\mathbf{z}}_i|\mathbf{w}_i)$, by using Equations (4.128) and (4.129), it follows that

$$\begin{aligned} P(\tilde{\mathbf{z}}^{\neg ih}|\mathbf{w}) &\propto P(\mathbf{w}|\tilde{\mathbf{z}}^{\neg ih}) P(\tilde{\mathbf{z}}^{\neg ih}) \\ &\propto \prod_{k=1}^{L} \frac{\prod_j \Gamma(\mathrm{F}_{j|k}^{\neg ih} + \beta)}{\Gamma(\mathrm{F}_{\cdot|k}^{\neg ih} + M\beta)} \times \prod_{i'=1}^{N} \frac{\prod_k \Gamma(\mathrm{f}_{k|i'}^{\neg ih} + \alpha)}{\Gamma(\mathrm{f}_{\cdot|i'}^{\neg ih} + L\alpha)}, \end{aligned} \tag{4.131}$$

where

$$\mathrm{F}_{j|k}^{\neg ih} = \sum_{i'=1}^{i-1} \mathrm{f}_{j|i'k} + \mathrm{f}_{j|ik}^{\neg h} + \sum_{i'=i+1}^{N} \mathrm{f}_{j|i'k}, \tag{4.132}$$

and

$$\mathrm{f}_{j|ik}^{\neg h} = \begin{cases} \mathrm{f}_{j|ik} - 1 & \text{if } k = \tilde{z}_{ih} \\ \mathrm{f}_{j|ik} & \text{if } k \neq \tilde{z}_{ih} \end{cases}. \tag{4.133}$$

Namely, $\mathrm{F}_{j|k}^{\neg ih}$ means the total frequency of t_j occurring as tokens allocated to topic k in D when token w_{ih} is removed. Similarly, $\mathrm{f}_{k|i'}^{\neg ih}$ indicates the total number of tokens allocated to topic k in

[49] The integral of a probabilistic distribution in Equation (4.127) becomes

$$\int P(\boldsymbol{\phi}_k|\beta)\, d\boldsymbol{\phi}_k = \frac{\Gamma(\sum_j \beta)}{\prod_j \Gamma(\beta)} \int \prod_j \phi_{j|k}^{\beta-1} d\boldsymbol{\phi}_k = 1,$$

which leads to an equation,

$$\int \prod_j \phi_{j|k}^{\beta-1} d\boldsymbol{\phi}_k = \frac{\prod_j \Gamma(\beta)}{\Gamma(\sum_j \beta)}.$$

If t_1 appears as a token at the position such that $\tilde{z}_{i2} = 3$ (i.e., $h = 2$ and $k = 3$), then a $\phi_{1|3}$ in Equation (4.125) is incorporated into the integral and $\phi_{1|3}^{\beta-1}$ changes to $\phi_{1|3}^{\beta}$. Thus by counting $\phi_{j|k}$ for all pairs of a topic and a term of all tokens in d_i, Equation (4.128) is obtained.

$d_{i'}$ in the case that token w_{ih} is ignored (inevitably, if $i' \neq i$, then $\mathrm{f}_{k|i'}$ does not take any influence). Needless to say, $\mathrm{F}_{\cdot|k}^{\neg ih} = \sum_j \mathrm{F}_{j|k}^{\neg ih}$ and $\mathrm{f}_{\cdot|i'}^{\neg ih} = \sum_k \mathrm{f}_{k|i'}^{\neg ih}$.

Equation (4.133) implies that $\mathrm{F}_{j|k}$ in $P(\tilde{z}_{ih} = k, \tilde{\mathbf{z}}^{\neg ih}|\mathbf{w})$ is just one larger than that in $P(\tilde{\mathbf{z}}^{\neg ih}|\mathbf{w})$ (i.e., $\mathrm{F}_{j|k} = \mathrm{F}_{j|k}^{\neg ih} + 1$ where $j = \omega(h|i)$ and $\tilde{z}_{ih} = k$), and that the other terms $\mathrm{F}_{j'|k'}$ where $j' \neq \omega(h|i)$ and $k' \neq k$ are completely identical with those in $P(\tilde{\mathbf{z}}^{\neg ih}|\mathbf{w})$, which are canceled out in the process of calculating Equation (4.130). So, the corresponding part in Equation (4.130) reduces to a simple form,

$$\frac{\Gamma(\mathrm{F}_{j|k}^{\neg ih} + 1 + \beta)}{\Gamma(\mathrm{F}_{j|k}^{\neg ih} + \beta)} = \mathrm{F}_{j|k}^{\neg ih} + \beta, \;\; j = \omega(h|i), \tag{4.134}$$

since generally $\Gamma(x+1)/\Gamma(x) = x$. Because the same operation can be applied to $\mathrm{F}_{\cdot|k}^{\neg ih}$, $\mathrm{f}_{k|i'}^{\neg ih}$ and $\mathrm{f}_{\cdot|i'}^{\neg ih}$, the full conditional probability in Equation (4.130) becomes finally

$$P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}^{\neg ih}, \mathbf{w}) \propto \frac{\mathrm{F}_{\omega(h|i)|k}^{\neg ih} + \beta}{\mathrm{F}_{\cdot|k}^{\neg ih} + M\beta} \frac{\mathrm{f}_{k|i}^{\neg ih} + \alpha}{\mathrm{f}_{\cdot|i}^{\neg ih} + L\alpha} \propto \frac{\mathrm{F}_{\omega(h|i)|k}^{\neg ih} + \beta}{\mathrm{F}_{\cdot|k}^{\neg ih} + M\beta} (\mathrm{f}_{k|i}^{\neg ih} + \alpha), \tag{4.135}$$

where $\omega(h|i) = j$ (note that factor $\mathrm{f}_{\cdot|i}^{\neg ih} + L\alpha$ is constant when assigning a topic to each token) [109]. Actually, a set of probabilities that

$$\pi_{ih}(k) = \frac{P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}^{\neg ih}, \mathbf{w})}{\sum_{k'=1}^{L} P(\tilde{z}_{ih} = k'|\tilde{\mathbf{z}}^{\neg ih}, \mathbf{w})}, \;\; k = 1, \ldots, L, \tag{4.136}$$

based on Equation (4.135) is used as a discrete distribution for randomly assigning a topic to token w_{ih} in s-th sample.

The parameters of LDA model can be empirically estimated by

$$\hat{\phi}_{j|k} = \frac{\mathrm{F}_{j|k} + \beta}{\mathrm{F}_{\cdot|k} + M\beta}, \tag{4.137}$$

and

$$\hat{\theta}_{ik} = \frac{\mathrm{f}_{k|i} + \alpha}{\mathrm{f}_{\cdot|i} + L\alpha}, \tag{4.138}$$

in each sample after assigning topics to all tokens. Therefore, it is easy to compute an average of the values in each sample such that

$$\bar{\hat{\phi}}_{j|k} = \frac{1}{r - B} \sum_{s=B+1}^{r} \hat{\phi}_{j|k}^{(s)}, \quad \bar{\hat{\theta}}_{ik} = \frac{1}{r - B} \sum_{s=B+1}^{r} \hat{\theta}_{ik}^{(s)},$$

for $k = 1, \ldots, L$, $j = 1, \ldots, M$ and $i = 1, \ldots, N$ (where $\hat{\phi}_{j|k}^{(s)}$ and $\hat{\theta}_{ik}^{(s)}$ are estimates in s-th iteration), and they can be used as estimates of the parameters.

Also, by using the estimates, document clusters can be generated as

$$C_k = \left\{ d_i \;\middle|\; \arg\max_{k'=1,\ldots,L} \bar{\hat{\theta}}_{ik'} \right\}. \tag{4.139}$$

Otherwise, it is possible to use a smaller subset of samples, or to count 'cluster patterns' without calculating $\hat{\theta}_{ik}$ for creating clusters as discussed in Section 4.1.2 ('cluster pattern count method').

Figure 4.13 shows the algorithm of clustering documents based on Gibbs sampling for the LDA model. Because $\phi_{j|k}$ and θ_{ik} do not have to be 'directly' estimated in the procedure, the algorithm becomes simple and compact. Basically, data to be recorded in the iterations are only values of $\mathrm{F}_{j|k}$ and $\mathrm{f}_{k|i}$, and it is enough to update them only when the current topic of a given token is changed. Although the total number of iterations r has to be large in a situation, the Gibbs sampling for LDA keeps its computational complexity at the level of $O(l \times L \times r)$, basically (l denotes a collection length as before).

‡ Document clustering based on Gibbs sampling for LDA model

Set: The number of clusters L, hyperparameters α and β, and the number of total iterations r (constant B is also needed if using Equation (4.139) as a clustering criterion).

1) After allocating randomly initial topics to all tokens, compute $\mathrm{F}_{j|k}$ and $\mathrm{f}_{i|k}$ ($k = 1, \ldots, L$; $j = 1, \ldots, M$; $i = 1, \ldots, N$), and set s to 1 (i.e., $s \leftarrow 1$).

2) For $i = 1, \ldots, N$, select randomly a topic to token w_{ih} based on Equation (4.136) and if the topic of w_{ih} is changed, then update $\mathrm{F}_{j|k}$ and $\mathrm{f}_{i|k}$ for old and new topics ($h = 1, \ldots, l_i$).

3) $s \leftarrow s + 1$. If $s > r$, then go to step 4). Otherwise, return to 2).

4) Terminate the procedure after allocating d_i ($i = 1, \ldots, N$) to a cluster based on Equation (4.139) or other methods.

Out: Clusters $C_1, \ldots, C_L$, and estimators of $\phi_{j|k}$ and θ_{ik} if needed.

Figure 4.13: Document clustering by based on Gibbs sampling for LDA mode

However, it should be noted that results from Gibbs sampling for the LDA model would be sensitive to values of hyperparameters α and β. As shown in Equation (4.135), the hyperparameter works actually as a smoothing factor in the process of Gibbs sampling. If the value is small, then it is relatively difficult that a token transfers to another topic, which means that the result from sampling becomes stable, and consequently, possibility of escaping from a local maximum becomes relatively lower. Inversely, when the hyperparameter is large, topic assignment is often unstable and it may be possible to obtain many inappropriate samples.

In Griffiths & Steyvers(2004) [109], it was recommended that $\alpha = 50/L$ and $\beta = 0.1$ for a set of scientific abstracts, whereas a more recent experiment [198] showed that a combination of $\alpha = 0.1$ and $\beta = 0.01$ works well for a set of news articles ($\beta = 0.01$ was originally used for a collection of text passages from educational materials in [294]). It may be naturally to consider that appropriate values of the hyperparameters are dependent on each situation to which this method is applied. For example, if 5% of all tokens included in a given document set is assumed to be distributed for smoothing each estimator of θ_{ik} and $\phi_{j|k}$ evenly, then it follows that

$$\alpha = \frac{l \times 0.05}{NL}, \tag{4.140}$$

because the number of parameters θ_{ki} is $N \times L$, and similarly,

$$\beta = \frac{l \times 0.05}{ML}. \tag{4.141}$$

Although '0.05' is an arbitrary value and there is no evidence on the validity, this would be a way for determining size of hyperparameters depending on a property of the target document set.

In the case of the sample DB, since $l = 62$, $M = 6$ and $N = 10$, it becomes that $\alpha = 0.103$ and $\beta = 0.172$ when $L = 3$. Table 4.17 shows parameters estimated from samples generated by a single chain of the Gibbs sampling when the algorithm in Figure 4.13 was applied to the sample DB with $L = 3$, $\alpha = 0.103$, $\beta = 0.172$, $r = 1100$ and $B = 100$. It seems that the clustering criterion in Equation (4.139) worked well in the experiment because a set of clusters $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$ was successfully generated. Especially, averages of $\hat{\theta}_{ik}$ of d_1, d_2 and d_3 for C_1 were over 0.9, which indicates that they were allocated to a particular topic very frequently in the iterative sampling and that a 'stable' result on the three documents was obtained. In contrast, when using larger values of α and β, the estimated values spread evenly over all topics due to the smoothing effect. For example, an execution of Gibbs sampling for the

Table 4.17: Examples of estimating LDA model by Gibbs sampling

	Averages of $\hat{\theta}_{ik}^{(s)}$				Averages of $\hat{\phi}_{j\|k}^{(s)}$		
	C_1	C_2	C_3		C_1	C_2	C_3
d_1	.934	.032	.033	t_1	.317	.005	.005
d_2	.916	.042	.041	t_2	.647	.055	.023
d_3	.950	.026	.023	t_3	.011	.541	.046
d_4	.024	.815	.160	t_4	.007	.187	.218
d_5	.309	.468	.222	t_5	.009	.082	.612
d_6	.107	.219	.674	t_6	.009	.128	.095
d_7	.088	.727	.184				
d_8	.032	.110	.857				
d_9	.027	.075	.897				
d_{10}	.030	.435	.535				

Note: 1.A single chain with $r = 1100$ and $B = 100$.
2.$\alpha = 0.172$ and $\beta = 0.103$.

sample DB with $\alpha = 2.0$ and $\beta = 2.0$ provided a result that the averages of $\hat{\theta}_{3k}$ for three topics in d_3 were .307, .361 and .332, respectively, from which it would be hard to obtain a valid clustering set.

It should be noted that the average of $\hat{\theta}_{52}$ (i.e., value of d_5 for C_2) is not large (= .468) even though using 0.103 and 0.172 as hyperparameters (see Table 4.17). Figure 4.14 shows frequency distributions of $\hat{\theta}_{31}^{(s)}$ (value of d_3 for C_1) and of $\hat{\theta}_{52}^{(s)}$ in the 1000 samples of this experiment, which implies that some 'bad' samples in which the value of t_5 for C_2 is small were drawn in the process of 1000 iterations. This suggests that topic assignment to tokens in d_5 is relatively difficult, and for safety, it may be better to use multiple sets of samples generated from parallel chains [50].

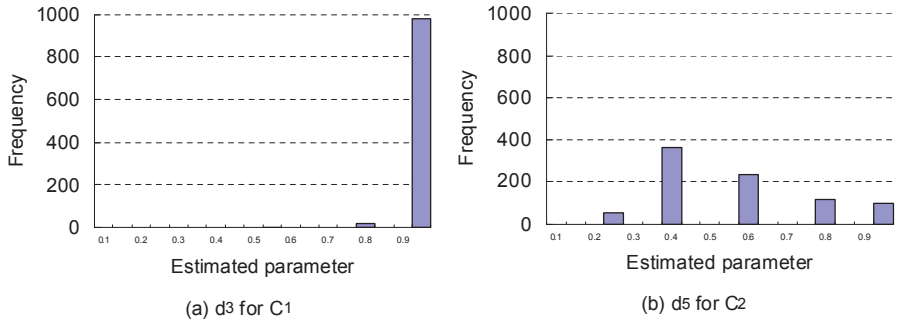


Figure 4.14: Distributions of estimated values $\hat{\theta}_{31}$ and $\hat{\theta}_{52}$ (in 1000 samples)

The result of executing 100 chains of Gibbs sampling for the sample DB is shown in Table 4.18, which indicates that 92% of chains generated a 'valid' set of $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7, d_{10}\}, \{d_6, d_8, d_9\}\}$ or $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9, d_{10}\}\}$ in the case that $\alpha = 0.103$, $\beta = 0.172$ and $r = 1100$ ($B = 100$). In contrast, chains with only 200 iterations sometimes generated 'invalid' samples. Similar trend was also observed when using $\alpha = 0.1$ and $\beta = 0.01$ adopted in an experiment by [198] (see Table 4.18).

[50] Actually, Griffiths & Steyvers(2004) [109] reported an experiment in which eight Markov chains were executed and 10 samples were taken from each chain with an interval of 100 iterations.

Table 4.18: Multiple 100 chains of Gibbs sampling for LDA in sample DB

		$\alpha = .100$		$\alpha = .172$	
		$\beta = .010$		$\beta = .103$	
	no. of iterations =	200	1100	200	1100
1.	$\{d_1, d_2, d_3, d_4, d_5, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$	9	1	0	0
2.	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7, d_{10}\}\{d_6, d_8, d_9\}$	8	5	8	1
3.	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7\}\{d_6, d_8, d_9, d_{10}\}$	0	1	1	0
4.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7, d_8, d_9\}\{d_{10}\}$	3	3	3	1
5.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7, d_9\}\{d_8, d_{10}\}$	1	0	1	0
6.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7, d_{10}\}\{d_8, d_9\}$	2	0	0	0
7.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_6, d_7\}\{d_8, d_9, d_{10}\}$	3	8	5	1
8.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7, d_{10}\}\{d_6, d_8, d_9\}$	29	15	33	18
9.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$	39	60	37	74
10.	$\{d_1, d_2, d_3\}\{d_4, d_7, d_8, d_{10}\}\{d_5, d_6, d_9\}$	0	2	0	1
11.	$\{d_1, d_2, d_3\}\{d_4, d_7, d_{10}\}\{d_5, d_6, d_8, d_9\}$	2	2	4	2
12.	$\{d_1, d_2, d_3\}\{d_4, d_8, d_9, d_{10}\}\{d_5, d_6, d_7\}$	1	0	1	0
13.	$\{d_1, d_2, d_3\}\{d_4, d_{10}\}\{d_5, d_6, d_7, d_8, d_9\}$	0	0	4	0
14.	Others	3	3	3	2
	Total	100	100	100	100

Note: 1.$B = 100$.
2.'Others' includes cluster sets appearing only a single chain.

4.1.5 Hierarchical Dirichlet process mixture model

Infinite mixture model

Like other techniques of partitioning a document set such as k-means, finite mixture model, PLSA and so on, it is necessary for executing LDA to determine the number of clusters a priori. In contrast, *hierarchical Dirichlet process* (HDP) mixture model that is an extension of the LDA can estimate automatically the number of latent topics, which is a useful property for DC. Namely, without any model selection process in which estimation is repeated with varying the number of topics and the log likelihood scores between them are compared, an optimal number can be empirically obtained when the HDP mixture model is used (actually, the resulting number may be more or less dependent on values of its hyperparameters as discussed later).

The reason why the number of topics L can be automatically estimated by the HDP mixture model is to assume that L is infinite (i.e., $L = \infty$) in the level of conceptual modeling, and actually, L converges to a finite value in a process of estimation from observed data [51]. More specifically, generation $\boldsymbol{\theta}_i \sim P_D(\cdot|\boldsymbol{\alpha})$ in the LDA model is replaced by $\boldsymbol{\theta}_i \sim P_D(\cdot|\alpha_0\boldsymbol{\rho})$ in the HDP mixture model, where $\boldsymbol{\rho}$ is theoretically assumed to be a vector with 'infinite' length (inevitably, $\boldsymbol{\theta}_i$ becomes also an infinite-length vector) and its k-th component is defined such that

$$\rho_k = \tilde{\rho}_k \prod_{k'=1}^{k-1} (1 - \tilde{\rho}_{k'}), \ k = 1, 2, \ldots, \infty, \tag{4.142}$$

where $0 < \tilde{\rho}_k < 1$ (note that α_0 is a constant). Particularly, $\tilde{\rho}_k$ is distributed according to a Beta distribution with parameters 1 and ζ [300, 302]. Equation (4.142) is often called 'stick-breaking' construction. By combining generation $\boldsymbol{\phi}_k \sim P_D(\cdot|\boldsymbol{\beta})$ with $\boldsymbol{\rho}$, a random measure G_0 can be defined such that

$$G_0 = \sum_{k=1}^{\infty} \rho_k \delta(\boldsymbol{\phi}_k), \tag{4.143}$$

[51] Strictly, the number of document clusters may be different from that of topics. But, for simplicity, L is used for mathematical explanation for a while. In Figure 4.15, the two numbers are represented by different notations (i.e., L and L').

where $\delta(x)$ means a distribution concentrated on a single point x. Actually, G_0 is a distribution of $\boldsymbol{\phi}_k$, which enables to select randomly a particular $\boldsymbol{\phi}_k$ from a set $\{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots\}$. Since Equation (4.143) is derived from a Dirichlet process (DP) into which $\boldsymbol{\beta}$ and ζ are incorporated as parameters, it is possible to interpret that the DP provides an infinite mixture model, which is often called *Dirichlet process mixture model*.

In particular, since $\boldsymbol{\theta}_i$ is generated from $P_D(\cdot|\alpha_0\boldsymbol{\rho})$ after $\boldsymbol{\rho}$ was obtained, it is feasible to write again that $G_i = \sum_{k=1}^{\infty} \theta_{ik}\delta(\boldsymbol{\phi}_k)$ for each document $(i = 1, \ldots, N)$. Because $\boldsymbol{\rho}$ is generated in the DP process about G_0, G_0 and G_i can be considered to form a hierarchy. Thus the model based on G_0 and G_i should be called hierarchical DP (HDP) mixture model, which was formalized by Teh, Jordan, Beal & Blei(2005, 2006) [300, 301]. In the basic procedure of HDP mixture model, after $\boldsymbol{\theta}_i$ is generated, topic assignment $\tilde{z}_{ih}$ is determined by $\pi(\cdot|\boldsymbol{\theta}_i)$, and then, a term is selected as token w_{ih} from a distribution $\pi(\cdot|\tilde{z}_{ih}, \boldsymbol{\phi})$, similarly to the LDA model.

Another method for realizing the DP is to formalize an infinite limit of the finite mixture model [219]. First, a conditional distribution

$$P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}_i^{\neg ih}) = P(\tilde{z}_{ih} = k, \tilde{\mathbf{z}}_i^{\neg ih})/P(\tilde{\mathbf{z}}_i^{\neg ih}), \tag{4.144}$$

is considered here. If $\boldsymbol{\rho} = [1/L, \ldots, 1/L]^T$ for a finite L, then it becomes that

$$P(\tilde{\mathbf{z}}_i) = \int P(\tilde{\mathbf{z}}_i|\boldsymbol{\theta}_i)P(\boldsymbol{\theta}_i|\alpha_0\boldsymbol{\rho})d\boldsymbol{\theta}_i \propto \frac{\prod_k \Gamma(\mathrm{f}_{k|i} + \alpha_0/L)}{\Gamma(\mathrm{f}_{\cdot|i} + \alpha_0)}, \tag{4.145}$$

from the observed data, since $P(\tilde{\mathbf{z}}_i|\boldsymbol{\theta}_i)$ is a discrete distribution and $P(\boldsymbol{\theta}_i|\alpha_0\boldsymbol{\rho})$ is a Dirichlet distribution (see Equation (4.129)). By applying Equation (4.145) to Equation (4.144), then $P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}_i^{\neg ih}) = (\mathrm{f}_{k|i}^{\neg ih} + \alpha_0/L)/(\mathrm{f}_{\cdot|i}^{\neg ih} + \alpha_0)$ is obtained as before. So, letting L go to infinity (i.e., $L \to \infty$) leads to a limit such that

$$P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}_i^{\neg ih}) \to \frac{\mathrm{f}_{k|i}^{\neg ih}}{l_i - 1 + \alpha_0}, \tag{4.146}$$

which implies that there is a probability of selecting a 'new' topic as a value of $\tilde{z}_{ih}$ because $\sum_k \mathrm{f}_{k|i}^{\neg ih} = l_i - 1$. Consequently, under assumptions of the DP mixture model, the probability of topic assignment becomes that

$$P(\tilde{z}_{ih} = k|\tilde{\mathbf{z}}_i^{\neg ih}) \to \begin{cases} \frac{\mathrm{f}_{k|i}^{\neg ih}}{l_i - 1 + \alpha_0}, & \text{if } k \text{ appears already in } \tilde{\mathbf{z}}_i^{\neg ih} \\ \frac{\alpha_0}{l_i - 1 + \alpha_0}, & \text{otherwise (i.e., 'new' topic)} \end{cases} \tag{4.147}$$

$(i = 1, \ldots, N;\ h = 1, \ldots, l_i;\ k = 1, 2, \ldots)$.

Estimation by Gibbs sampling

Teh, Jordan, Beal & Blei(2005) [300, 301] provided a Gibbs sampling technique for estimating parameters in the HDP mixture model based on a scheme of the '*Chinese restaurant franchise*'(CRF), in which each document is regarded as a single Chinese restaurant having tables with customers eating a dish [52]. The dish corresponds to a topic (i.e., a cluster in DC), which is assumed to be selected from a common menu of the franchise. A sequence number of a dish selected at u-th table in restaurant d_i is denoted by k_{iu}. For example, if customers at the second table in d_5 eat the third dish in the common menu, then $k_{52} = 3$. Note that only a single dish is eaten at each table, but the dish may be exchanged with another dish (e.g., from $k_{52} = 3$ to $k_{52} = 1$). Since a customer means a word token, it can be interpreted that k_{iu}-th topic is assigned to customer w_{ih} sitting at u-th table, and the term at position of token w_{ih} is assumed to be generated from $\boldsymbol{\phi}_k$ where $k = k_{iu}$. When the dish is exchanged, the topic of all customers (i.e., tokens) at the table is shifted concurrently. Also, it is possible that the customer transfers to another table in the restaurant.

Namely, in the CRF model, a value of $\tilde{z}_{ih}$ (i.e., a topic) will be determined through the table to which w_{ih} belongs rather than directly drawn from $\pi(\cdot|\boldsymbol{\theta}_i)$. Finally, a term in d_i is supposed to

[52] The number of customers at each table in a single restaurant can be formalized as *Chinese restaurant process* (CRP), which describes a distribution of partitions of n integers $\{1, 2, \ldots, n\}$ where n is discrete time index [296]. In the case of HDP, a set of restaurants administrated as a 'franchise' chain is considered.

be drawn from $\pi(\cdot|u_{ih}, \mathbf{k}_i, \boldsymbol{\phi})$ where u_{ih} denotes a sequence number of a table selected by customer w_{ih}, $\mathbf{k}_i = [k_{i1}, \ldots, k_{im_{i\cdot}}]^T$, and $m_{i\cdot}$ indicates the number of tables in d_i. If u_{ih} is generated from a discrete distribution $\pi(\cdot|\tilde{\boldsymbol{\theta}}_i)$ where $\tilde{\boldsymbol{\theta}}_i$ is an $m_{i\cdot}$-dimensional vector following a Dirichlet distribution $P_D(\cdot|\alpha_0/m_{i\cdot}, \ldots, \alpha_0/m_{i\cdot})$, then its limit with $m_{i\cdot} \to \infty$ becomes a distribution similar to Equation (4.147), which can be written as

$$P(u_{ih} = u|\mathbf{u}_i^{\neg ih}) \to \begin{cases} \frac{\mathrm{f}_{u|i}^{\neg ih}}{l_i - 1 + \alpha_0}, & \text{if } u \text{ appears already in } \mathbf{u}_i^{\neg ih} \\ \frac{\alpha_0}{l_i - 1 + \alpha_0}, & u \text{ is a 'new' table} \end{cases}, \tag{4.148}$$

where $\mathbf{u}_i = [u_{i1}, \ldots, u_{il_i}]^T$ and $\mathrm{f}_{u|i}^{\neg ih}$ denotes the times that u-th table appears in $\mathbf{u}_i^{\neg ih}$.

The Gibbs sampling based on the the CRF consists of (a) drawing tables and (b) drawing dishes. The conditional probability of drawing a table for token w_{ih} can be written as

$$\begin{aligned} P(u_{ih} = u|\mathbf{u}^{\neg ih}, \mathbf{w}, \mathbf{k}) &= \frac{P(u_{ih} = u, \mathbf{u}^{\neg ih}|\mathbf{w}, \mathbf{k})}{P(\mathbf{u}^{\neg ih}|\mathbf{w}, \mathbf{k})} \\ &= \frac{P(\mathbf{w}|u_{ih} = u, \mathbf{u}^{\neg ih}, \mathbf{k})}{P(\mathbf{w}|\mathbf{u}^{\neg ih}, \mathbf{k})} \times \frac{P(u_{ih} = u, \mathbf{u}^{\neg ih}|\mathbf{k})}{P(\mathbf{u}^{\neg ih}|\mathbf{k})}, \end{aligned} \tag{4.149}$$

similarly to Equation (4.130) in the LDA where $\mathbf{u} = [\mathbf{u}_1^T, \ldots, \mathbf{u}_N^T]^T$ and $\mathbf{k} = [\mathbf{k}_1^T, \ldots, \mathbf{k}_N^T]^T$. If u-th table has been previously used (i.e., not a 'new' table), then particular $\boldsymbol{\phi}_k$ can be specified for the table and the first term of the most right-hand side in Equation (4.149) is written as

$$\frac{P(\mathrm{w}_{ih} = t_j, \mathbf{w}^{\neg ih}|u_{ih} = u, \mathbf{u}^{\neg ih}, \mathbf{k})}{P(\mathrm{w}_{ih} = t_j, \mathbf{w}^{\neg ih}|\mathbf{u}^{\neg ih}, \mathbf{k})} = \frac{\mathrm{F}_{j|k[ih]}^{\neg ih} + \beta}{\mathrm{F}_{\cdot|k[ih]}^{\neg ih} + M\beta} \equiv f_k^{\neg ih}(\mathrm{w}_{ih}), \tag{4.150}$$

by applying the analytical calculations deriving Equation (4.134) again (note that $k[ih]$ in the equation denotes a sequence number of the dish eaten at the table to which w_{ih} belongs, i.e., $k[ih] \equiv k_{iu_{ih}}$, and Dirichlet distributions $P_D(\boldsymbol{\phi}_k|\boldsymbol{\beta})$ for $k = 1, \ldots, L$ are assumed to be a symmetric again with parameter β).

In contrast, because particular $\boldsymbol{\phi}_k$ is not determined for a new table (which is denoted by $u^\dagger$), $P(\mathbf{w}|u_{ih} = u^\dagger, \mathbf{u}^{\neg ih}, \mathbf{k})$ has to be computed with considering that all dishes (i.e., topics) have a chance to be selected for the table. A dish of a table is drawn from a discrete distribution $\pi(\cdot|\boldsymbol{\kappa})$, and $\boldsymbol{\kappa}$ is also generated from a Dirichlet distribution $P_D(\cdot|\,\xi/L, \ldots, \xi/L)$ in the CRF model [300], namely, $k_{iu} \sim \pi(\cdot|\boldsymbol{\kappa})$ and $\boldsymbol{\kappa} \sim P_D(\cdot|\,\xi/L, \ldots, \xi/L)$. Under the assumption, the probability that topic τ_k is selected for u-th table in d_i given data $\mathbf{k}^{\neg iu}$ becomes again

$$P(k_{iu} = k|\mathbf{k}^{\neg iu}) \to \begin{cases} m_{\cdot k}/(m_{\cdot\cdot} + \xi), & \text{if } k = 1, \ldots, L \\ \xi/(m_{\cdot\cdot} + \xi), & \text{otherwise ('new' topic)} \end{cases}, \tag{4.151}$$

with $L \to \infty$, where $m_{\cdot k}$ is the number of tables eating k-th dish in all restaurants and $m_{\cdot\cdot} = \sum_{k=1}^{L} m_{\cdot k}$ (note that $u^\dagger = m_{i\cdot} + 1$). Since $P(\mathbf{w}|u_{ih} = u^\dagger, \mathbf{u}^{\neg ih}, \mathbf{k})$ should be formulated as an expectation by the probabilities of selecting a dish, Equation (4.150) for new table $u^\dagger$ becomes that

$$\begin{aligned} \frac{P(\mathrm{w}_{ih} = t_j, \mathbf{w}^{\neg ih}|u_{ih} = u^\dagger, \mathbf{u}^{\neg ih}, \mathbf{k})}{P(\mathrm{w}_{ih} = t_j, \mathbf{w}^{\neg ih}|\mathbf{u}^{\neg ih}, \mathbf{k})} &= \sum_{k=1}^{L} \frac{m_{\cdot k}}{m_{\cdot\cdot} + \xi} f_k^{\neg ih}(\mathrm{w}_{ih} = t_j) + \frac{\xi}{m_{\cdot\cdot} + \xi} \frac{1}{M} \\ &\equiv P(\mathrm{w}_{ih} = t_j|\dagger), \end{aligned} \tag{4.152}$$

where $1/M$ is a priori distribution of selecting t_j. This means that a term is randomly selected from the vocabulary for the new topic because there is no prior knowledge about term distribution of a new dish in the franchise.

As to the second term of the most right-hand side in Equation (4.149), Equation (4.148) can be straightforwardly used since $P(u_{ih} = u, \mathbf{u}^{\neg ih}|\mathbf{k})/P(\mathbf{u}^{\neg ih}|\mathbf{k}) = P(u_{ih} = u|\mathbf{u}^{\neg ih}, \mathbf{k})$, (note that the selection of a table is independent of $\mathbf{k}$). Therefore, by combining Equation (4.148), (4.150) and (4.152), the conditional distribution of u_{ih} in Equation (4.149) can be computed as

$$P(u_{ih} = u|\mathbf{u}^{\neg ih}, \mathbf{w}, \mathbf{k}) \propto \begin{cases} \mathrm{f}_{u|i}^{\neg ih} f_k^{\neg ih}(\mathrm{w}_{ih}), & \text{if } u \text{ is previously used} \\ \alpha_0 P(\mathrm{w}_{ih} = t_j|\dagger), & \text{if } u = u^\dagger \end{cases}. \tag{4.153}$$

Consequently, in the case that $u \neq u^{\dagger}$, a probability distribution similar to Equation (4.135) in the LDA model was derived under some assumptions of the HDP mixture model. If a new table was drawn from Equation (4.153), then a dish (i.e., topic) of the new table for w_{ih} has to be successively drawn according to the probability distribution,

$$P(k_{iu^{\dagger}} = k|\mathbf{u}, \mathbf{k}^{\neg iu^{\dagger}}, \mathbf{w}) \propto \begin{cases} m_{\cdot k} f_k^{\neg ih}(\mathrm{w}_{ih}), & \text{if } k = 1, \ldots, L \\ \xi/M, & \text{if } k = k^{\dagger} \end{cases}, \tag{4.154}$$

where $k^{\dagger}$ denotes a new dish of the franchise (i.e., $k^{\dagger} = L + 1$) [53].

For sampling a dish for every table at next stage, it is necessary to consider the conditional probability for all tokens belonging to the target table because all customers (i.e., tokens) at the table eat the same dish as noted above. A set of index numbers of terms appearing as such tokens is denoted by $\Lambda[i, u]$ here. For example, if tokens w_{14} and w_{17} belong to the second table in d_1 and their terms are t_3 and t_6, respectively, then $\Lambda[1, 2] = \{3, 6\}$. From similar calculations, the likelihood corresponding to Equation (4.150) becomes that

$$\frac{P(\mathbf{w}|\mathbf{u}, k_{iu} = k, \mathbf{k}^{\neg iu})}{P(\mathbf{w}|\mathbf{u}, \mathbf{k}^{\neg iu})} = \frac{\prod_{j \in \Lambda[i,u]} \Gamma(\mathrm{F}_{j|k} + \xi)}{\prod_{j \in \Lambda[i,u]} \Gamma(\mathrm{F}_{j|k}^{\neg iu} + \xi)} \times \frac{\Gamma(\mathrm{F}_{\cdot|k}^{\neg iu} + L\xi)}{\Gamma(\mathrm{F}_{\cdot|k} + L\xi)} \equiv g_k^{\neg iu}(\mathbf{w}), \tag{4.155}$$

where $\mathrm{F}_{j|k}^{\neg iu}$ denotes occurrence frequency of t_j at tables eating k-th dish in all restaurants except u-th table in d_i and $\mathrm{F}_{\cdot|k}^{\neg iu} = \sum_j \mathrm{F}_{j|k}^{\neg iu}$. By using again Equation (4.151), a probability distribution for sampling topic k at each table,

$$P(k_{iu} = k|\mathbf{u}, \mathbf{k}^{\neg iu}, \mathbf{w}) \propto \begin{cases} m_{\cdot k}^{\neg iu} g_k^{\neg iu}(\mathbf{w}), & \text{if } k = 1, \ldots, L \\ \xi g_{k^{\dagger}}^{\neg iu}(\mathbf{w}), & \text{if } k = k^{\dagger} \end{cases}, \tag{4.156}$$

is obtained where $m_{\cdot k}^{\neg iu}$ denotes the number of tables eating k-th dish except u-th table in d_i and $g_{k^{\dagger}}^{\neg iu}(\mathbf{w})$ is a prior distribution. The prior can be reasonably assumed to be

$$g_{k^{\dagger}}^{\neg iu}(\mathbf{w}) = (1/M)^{|\Lambda[i,u]|}, \tag{4.157}$$

if terms at the table are randomly selected from the whole vocabulary in a situation that there is no knowledge [54].

After sampling repeatedly a table for each customer (i.e., token) and a dish (i.e., topic) for each table, it is possible to generate document clusters by counting the number of tokens at tables with k-th dish in d_i (i.e., $\mathrm{f}_{k|i}$) [55]. For example, by using the result, $\hat{\theta}_{ik}$ in Equation (4.138) can be estimated. It should be noted that the number of document clusters computed based on $\hat{\theta}_{ik}$ is not necessarily equal to the number of dishes (i.e., topics) because some dishes may not be dominant for any restaurant (i.e., document). Namely, the number of document clusters generated posteriorly may become less than that of topics in each sample. The number of topics is denoted by L' for discriminating it from the number of document clusters L afterward (i.e., $L' \geq L$).

An example of DC procedure based on the HDP mixture model is shown in Figure 4.15. As indicated in the figure, the number of topics L' is changed in the sampling process, and a value of L' is determined for each sample. If the set of clusters is produced by each sample (e.g., document d_i is allocated to topic $k = \arg\max_{k'} \mathrm{f}_{k'|i}$), then an empirical distribution of the number of document clusters L over samples can be observed.

Figure 4.16 shows the empirical distributions obtained by Gibbs sampling for the sample DB with $\alpha_0 = 0.1$ and $\beta = 0.01$. The number of iterations was 1100 times (a single chain), and the

[53] The distribution can be derived in a similar way for Equation (4.153). It follows that

$$P(k_{iu} = k|\mathbf{u}, \mathbf{k}^{\neg iu}, \mathbf{w}) = \frac{P(\mathbf{k}|\mathbf{u}, \mathbf{w})}{P(\mathbf{k}^{\neg iu}|\mathbf{u}, \mathbf{w})} = \frac{P(\mathbf{w}|\mathbf{k}, \mathbf{u})}{P(\mathbf{w}|\mathbf{k}^{\neg iu}, \mathbf{u})} \times \frac{P(\mathbf{k}|\mathbf{u})}{P(\mathbf{k}^{\neg iu}|\mathbf{u})},$$

and $P(\mathbf{w}|\mathbf{k}, \mathbf{u})/P(\mathbf{w}|\mathbf{k}^{\neg iu}, \mathbf{u})$ is computed by $f_k^{\neg ih}(\mathrm{w}_{ih})$ in Equation (4.150) if $k \neq k^{\dagger}$. When $k = k^{\dagger}$, the probability can be assumed to be $1/M$ as before. Also, $P(\mathbf{k}|\mathbf{u})/P(\mathbf{k}^{\neg iu}|\mathbf{u})$ is reduced to Equation (4.151) because selection of a dish is not dependent on $\mathbf{u}$.

[54] Note that $|\Lambda[i, u]|$ indicates the number of elements in $\Lambda[i, u]$.

[55] In the CRF model, $\mathrm{f}_{k|i}$ is computed from $\mathrm{f}_{u|i}$ of all tables ordering k-th dish in d_i.

‡ Example of document clustering by Gibbs sampling for HDP mixture model

Set: Hyperparameters α_0, β and ξ, and the total number of iterations r.

1) Set the number of topics to an initial value (e.g., $L' = 2$) and assign each topic to an initial table for d_i ($m_{i\cdot} = L'$; $i = 1, \ldots, N$). After selecting randomly a table for every token in d_i ($i = 1, \ldots, N$), compute $\mathrm{F}_{j|k}$ and $\mathrm{f}_{u|i}$ ($k = 1, \ldots, L'$; $j = 1, \ldots, M$; $u = 1, \ldots, m_{i\cdot}$; $i = 1, \ldots, N$), and set s to 1 (i.e., $s \leftarrow 1$).

2) [Sampling of tables] For $i = 1, \ldots, N$, select randomly a table of token w_{ih} based on Equation (4.153) ($h = 1, \ldots, l_i$).

   * If a new table is added (i.e., $m_{i\cdot} \leftarrow m_{i\cdot} + 1$), then determine a topic for it based on Equation (4.154). Also, if a new topic is selected at this time, then $L' \leftarrow L' + 1$.
   * If a new table is added or token w_{ih} ($= t_j$) transfers to another existing table, then update $\mathrm{F}_{j|k}$ and $\mathrm{f}_{u|i}$ for old and new tables.
   * If the old table becomes empty as a result of the transfer, then it is removed (i.e., $m_{i\cdot} \leftarrow m_{i\cdot} - 1$). Also, if the number of tables belonging to the topic becomes zero by the removal, then the topic is deleted (i.e., $L' \leftarrow L' - 1$).

3) [Sampling of topics] For $i = 1, \ldots, N$, select randomly a topic of each table in d_i based on Equation (4.156) ($u = 1, \ldots, m_{i\cdot}$).

   * If a new topic is assigned or the topic is changed, then update $\mathrm{F}_{j|k}$ for old and new topics where $j \in \Lambda[i, u]$ (in the case of a new topic, $L' \leftarrow L' + 1$).
   * If the number of tables belonging to the old topic becomes zero at this time, then the topic is deleted (i.e., $L' \leftarrow L' - 1$).

4) $s \leftarrow s + 1$. If $s > r$, then go to step 5). Otherwise, return to 2).

5) Terminate the procedure after allocating d_i ($i = 1, \ldots, N$) to a cluster based on a method (e.g., by Equation (4.139)).

Out: L' and L, and clusters $C_1, \ldots, C_L$.

Figure 4.15: An example of DC procedure based on Gibbs sampling for HDP model

distributions in Figure 4.16 were complied from a set of 1000 samples after the burn-in-period (i.e., $B = 100$) with $\xi = 0.1$ and $\xi = 0.5$, respectively. It seems that $L = 4$ was obtained as an optimal number of document clusters in this experiment. However, since hyperparameter ξ governs a possibility that a new topic $k^{\dagger}$ is selected in the stage of sampling topics for tables, a bigger value of ξ tends to generate more topics as the graphs show. Also, hyperparameters α_0 and β have an effect on the resulting numbers of topics and clusters. Actually, it would not be easy to determine optimal values of the hyperparameters although the HDP mixture model has an advantage of determining automatically the numbers of topics and clusters for observed data.

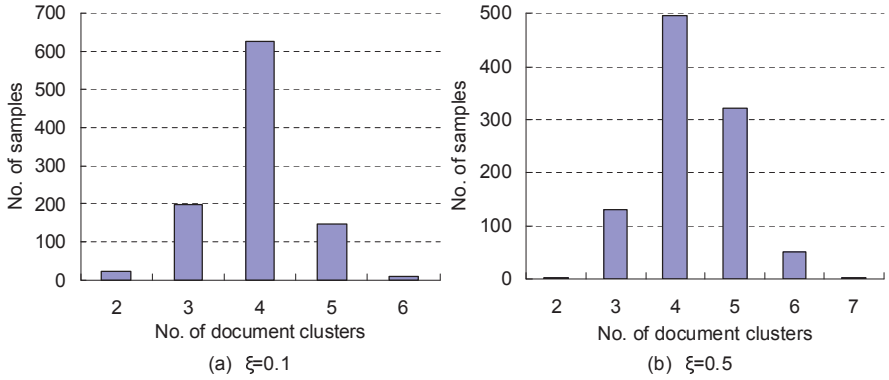


Figure 4.16: Numbers of document clusters estimated by HDP for sample DB (for 1000 samples after burn-in-period when $\alpha_0 = 0.1$ and $\beta = 0.01$)

Since a topic of all terms at a table may be changed together in the stage of sampling the topics, label switching would naturally occur. For example, in the experiment using the sample DB, d_1, d_2, d_3, d_4 and d_5 were allocated to topics such that

d_1: $k = 1$, d_2: $k = 1$, d_3: $k = 1$, d_4: $k = 3$, d_5: $k = 3$ in 515th sample,

d_1: $k = 3$, d_2: $k = 3$, d_3: $k = 3$, d_4: $k = 1$, d_5: $k = 1$ in 570th sample,

according to the rule that $k = \arg\max_{k'} \mathrm{f}_{k'|i}$. Needless to say, the two samples indicate the same clustering result of $\{d_1, d_2, d_3\}$ and $\{d_4, d_5\}$ even though the topic numbers are swapped. Therefore, a criterion corresponding to Equation (4.75) is required for determining exactly a topic for each document. However, if disregarding estimation of $\phi_{j|k}$ and θ_{ki} and trying only production of document clusters, then it is enough to apply the 'cluster pattern count method' described in Section 4.1.2. The result of applying it to the data of samples in Figure 4.16(a) is shown in Table 4.19, which indicates that the set of documents, $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_7\}, \{d_6, d_8, d_9\}, \{d_{10}\}\}$ ($L = 4$), was appearing most frequently in a single chain with $\xi = 0.1$.

Teh, Jordan, Beal & Blei(2005) [300] described other methods of Gibbs sampling based on the CRF model. Alternatively, a variational inference technique for estimation of the HDP mixture model was also proposed [302]. Instead of the algorithm in Figure 4.15, it is possible to use them for DC based on the HDP mixture model. In addition, the Chinese restaurant model was applied to construct a hierarchical topic structure [28].

4.2 Matrix-based Document Clustering

By manipulating a document-by-term matrix or term-by-document matrix like the sample DB in Table 3.1, based on the linear algebra theory, a partition of documents or terms can be obtained. In this section, DC techniques utilizing the matrix operations are reviewed.

4.2.1 Use of latent semantic indexing (LSI)

Latent semantic indexing (LSI) developed by Deerwester et al.(1990) [79] for enhancing effectiveness of IR tries to reduce dimensionality of a document-by-term matrix (or its transposed matrix) by using matrix operations. A document-by-term weight matrix is denoted by $\mathbf{W} = [w_{ij}]$ ($i =$

Table 4.19: Result of Gibbs sampling for HDP mixture model in sample DB

	Clusters	L	No. of samples
1.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$	4	276
2.	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$	4	194
3.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$	3	68
4.	$\{d_1\}\{d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$	5	66
5.	$\{d_1, d_2, d_3, d_5\}\{d_4, d_7\}\{d_6, d_8, d_9, d_{10}\}$	3	55
6.	$\{d_1, d_2, d_3\}\{d_4, d_7\}\{d_5, d_6, d_8, d_9\}\{d_{10}\}$	4	44
7.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7, d_{10}\}\{d_6, d_8, d_9\}$	3	35
8.	$\{d_1, d_2, d_3\}\{d_4, d_5, d_7\}\{d_6, d_9\}\{d_8, d_{10}\}$	4	21
9.	$\{d_1, d_2, d_3, d_4, d_5, d_7\}\{d_6, d_8, d_9, d_{10}\}$	2	20
10.	$\{d_1\}\{d_2, d_3, d_5\}\{d_4, d_7\}\{d_6, d_8, d_9\}\{d_{10}\}$	5	13
11.	Others: 86 sets of clusters in total	-	208
	Total	-	1000
Note:	A single chain with $\alpha_0 = 0.1$, $\beta = 0.01$ and $\xi = 0.1$.		

$1, \ldots, N; j = 1, \ldots, M$). According to matrix algebra, $\mathbf{W} = [w_{ij}]$ can be exactly decomposed such that

$$\mathbf{W} = \mathbf{UQV}^T, \tag{4.158}$$

where $\mathbf{U}$ is an $N \times r$ orthogonal matrix, $\mathbf{Q}$ is an $r \times r$ diagonal matrix, $\mathbf{V}$ is an $M \times r$ orthogonal matrix, and r indicates the 'rank' of matrix $\mathbf{W}$ (where $r \leq \min(N, M)$) [56]. Alternatively, Equation (4.158) can be represented as

$$\mathbf{W} = \eta_1 \mathbf{u}_1 \mathbf{v}_1^T + \eta_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \eta_r \mathbf{u}_r \mathbf{v}_r^T, \tag{4.159}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_r]$, $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_r]$, and $\eta_1, \ldots, \eta_r$ are diagonal elements of $\mathbf{Q}$. Equation (4.158) or (4.159) is called *singular value decomposition* (SVD). If $\eta_1 > \ldots > \eta_r$, then $\mathbf{u}_1$ and $\mathbf{v}_1$ are the dominant factors explaining data matrix $\mathbf{W}$, followed by $\mathbf{u}_2$ and $\mathbf{v}_2$, $\mathbf{u}_3$ and $\mathbf{v}_3$, and so on (see F

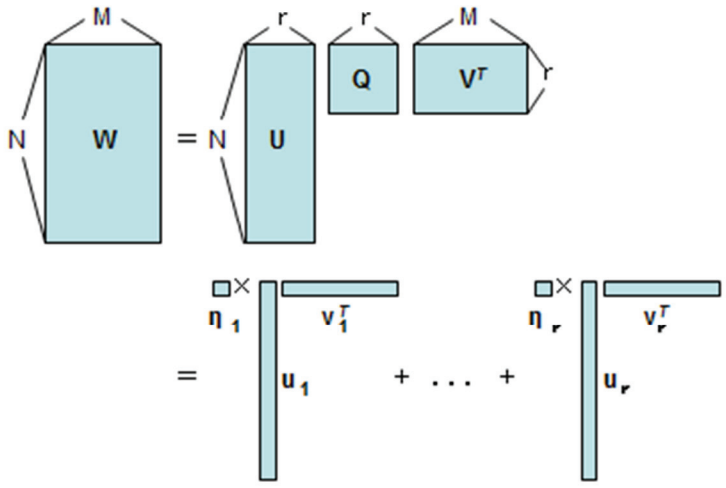


Figure 4.17: Singular value decomposition (SVD)

As an example, let $\mathbf{W}$ be a 5×6 matrix constructed by extracting only d_1, d_2, d_4 and d_8 from

[56] In this section, r denotes rank of a matrix, not the total number of iterations.

the sample DB and by adding specially $\mathbf{d}_{11} = [0, 0, 0, 0, 1, 4]^T$, namely,

$$\mathbf{W} = [\mathbf{d}_1, \mathbf{d}_2, \mathbf{d}_4, \mathbf{d}_8, \mathbf{d}_{11}]^T = \begin{bmatrix} 4 & 1 & 0 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 4 \end{bmatrix}, \tag{4.160}$$

which is decomposed into

$$\mathbf{U} = \begin{bmatrix} 0.000 & 0.851 & 0.000 & 0.000 & -0.526 \\ 0.000 & 0.526 & 0.000 & 0.000 & 0.851 \\ 0.911 & 0.000 & -0.237 & 0.338 & 0.000 \\ 0.397 & 0.000 & 0.280 & -0.874 & 0.000 \\ 0.112 & 0.000 & 0.930 & 0.349 & 0.000 \end{bmatrix}, \tag{4.161}$$

$$\mathbf{Q} = \begin{bmatrix} 5.255 & & & & \\ & 4.618 & & & \\ & & 4.231 & & \\ & & & 3.079 & \\ & & & & 2.382 \end{bmatrix}, \tag{4.162}$$

and

$$\mathbf{V} = \begin{bmatrix} 0.000 & 0.851 & 0.000 & 0.000 & -0.526 \\ 0.000 & 0.526 & 0.000 & 0.000 & 0.851 \\ 0.693 & 0.000 & -0.224 & 0.439 & 0.000 \\ 0.671 & 0.000 & -0.035 & -0.238 & 0.000 \\ 0.248 & 0.000 & 0.418 & -0.738 & 0.000 \\ 0.085 & 0.000 & 0.880 & 0.453 & 0.000 \end{bmatrix}, \tag{4.163}$$

since the rank of $\mathbf{W}$ is five (i.e, $r = 5$) [57]. It should be noted that $\mathbf{UQV}^T$ is also represented as $(-\mathbf{U})\mathbf{Q}(-\mathbf{V}^T)$, which is clear from Equation (4.159).

If $\mathbf{Q}$ is replaced by $\mathbf{Q}'$ whose diagonal elements are 5.255, 4.618, 0.000, 0.000 and 0.000 (i.e., new diagonal matrix $\mathbf{Q}'$ is constructed by extracting only the two largest elements from $\mathbf{Q}$, which can be represented as $\mathbf{Q}' = \text{diag}\,[5.255, 4.618, 0.000, 0.000, 0.000]$), then

$$\mathbf{W}' = \mathbf{UQ}'\mathbf{V}^T = \begin{bmatrix} 3.342 & 2.065 & 0.000 & 0.000 & 0.000 & 0.000 \\ 2.065 & 1.276 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 3.319 & 3.213 & 1.187 & 0.409 \\ 0.000 & 0.000 & 1.447 & 1.400 & 0.517 & 0.178 \\ 0.000 & 0.000 & 0.409 & 0.396 & 0.146 & 0.050 \end{bmatrix}. \tag{4.164}$$

Matrix $\mathbf{W}'$ can be interpreted as a new document-by-term matrix reflecting only two dominant dimensions within the original space represented by $\mathbf{W}$. In IR, the dominant dimension is often regarded as a 'major' concept represented by a set of different terms in documents. For example, the first dimension corresponding to 5.255 in $\mathbf{Q}$ would be a concept relating to third, fourth, fifth and sixth terms in $\mathbf{W}$ because values of these terms in the first column of $\mathbf{V}$ in Equation (4.163) are not zero. Actually, whereas a standard similarity measure between d_4 and d_{11} becomes zero by computation using the original matrix $\mathbf{W}$ due to no common term between them, the similarity of them is not zero for $\mathbf{W}'$ in Equation (4.164).

It is the main idea of LSI to extract such 'latent' concepts behind terms by using SVD of the document-by-term (or term-by-document) matrix, and if a search query is incorporated by a method into the LSI space constructed from some dominant dimensions, then it is expected that IR performance is improved. Otherwise, a 5×2 matrix,

$$\mathbf{UQ}' = \begin{bmatrix} 0.000 & 3.928 \\ 0.000 & 2.428 \\ 4.787 & 0.000 \\ 2.086 & 0.000 \\ 0.590 & 0.000 \end{bmatrix}, \tag{4.165}$$

[57] Because $N = 5$ and $M = 6$, $\min(N, M) = 5$. There is no row represented by a linear combination of other rows in $\mathbf{W}$, and therefore, straightforwardly $r = 5$. For details on the rank, see a textbook on matrix algebra.

can be potentially used for some applications as more compact representations of documents. Since $\mathbf{UQ} = \mathbf{WV}$ and $[\mathbf{WV}]^T = \mathbf{V}^T\mathbf{W}^T$, if $m < r \leq M$, the set of vectors $\mathbf{v}_1, ..., \mathbf{v}_m$ can be used for converting each document vector such that

$$\bar{\mathbf{d}}_i = [\mathbf{v}_1, ..., \mathbf{v}_m]^T \mathbf{d}_i, \ i = 1, \ldots, N. \tag{4.166}$$

The $\bar{\mathbf{d}}_i$ is an m-dimensional vector, which means that the LSI allows for reducing dimensions of a given document space with keeping main concepts (or semantics) in the resulting compact space.

In this trivial case, $\mathbf{UQ}'$ (or $\mathbf{U}$) can be used for DC. Equation (4.165) explicitly shows that the document set can be divided into two clusters, $C_1 = \{d_4, d_8, d_{11}\}$ and $C_2 = \{d_1, d_2\}$. However, in general, it is hard to use directly results from the SVD for DC. For example, in the case of the sample DB,

$$\mathbf{Q} = \text{diag}[\ 9.025, \quad 7.294, \quad 6.316, \quad 4.854, \quad 3.633, \quad 2.588\ ], \tag{4.167}$$

and

$$\mathbf{U} = \begin{bmatrix} -0.079 & 0.281 & 0.098 & 0.175 & -0.917 & -0.119 \\ -0.140 & 0.382 & 0.104 & 0.090 & 0.027 & 0.061 \\ -0.271 & 0.721 & 0.192 & 0.141 & 0.307 & 0.160 \\ -0.309 & -0.066 & -0.595 & 0.041 & -0.135 & 0.620 \\ -0.271 & 0.181 & -0.102 & -0.168 & 0.117 & -0.058 \\ -0.488 & -0.162 & 0.327 & -0.263 & 0.008 & -0.199 \\ -0.401 & 0.071 & -0.524 & -0.294 & -0.018 & -0.491 \\ -0.302 & -0.227 & 0.181 & 0.124 & -0.045 & 0.415 \\ -0.425 & -0.310 & 0.385 & -0.078 & -0.062 & 0.075 \\ -0.246 & -0.198 & -0.125 & 0.857 & 0.160 & -0.332 \end{bmatrix}, \tag{4.168}$$

from which it is not easy to extract any interpretation on clustering. As discussed later, $\mathbf{W}$ should be converted before executing SVD.

4.2.2 Principal component analysis (PCA)

By regarding each term as a variable, it is possible to treat $\mathbf{W}$ as a data matrix to which *principal component analysis* (PCA) is applied. In the PCA, a variable ζ is assumed to be constructed as a linear combination of observed variables $t_1, \ldots, t_M$ such that

$$\zeta = a_1 t_1 + a_2 t_2 + ... + a_M t_M, \tag{4.169}$$

where a_j is a weight ($j = 1, \ldots, M$) and $\sum_j a_j^2 = 1$. If observed variables and weights are written as vectors, $\mathbf{x} = [t_1, ..., t_M]^T$ and $\mathbf{a} = [a_1, ..., a_M]^T$, respectively, the constructed variable becomes $\zeta = \mathbf{a}^T\mathbf{x}$. By using a covariance matrix

$$\mathbf{\Sigma} = E[(\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^T], \tag{4.170}$$

the variance of ζ can be represented as $V(\zeta) = \mathbf{a}^T\mathbf{\Sigma}\mathbf{a}$ [58].

In the PCA, the weight vector $\mathbf{a}$ is empirically determined from observed data so that $V(\zeta)$ is maximized under a condition that $\mathbf{a}^T\mathbf{a} = 1$. For the maximization, it is necessary to differentiate

$$\mathcal{H} = \mathbf{a}^T\mathbf{\Sigma}\mathbf{a} - \lambda(\mathbf{a}^T\mathbf{a} - 1), \tag{4.171}$$

where λ is the Lagrange multiplier, and to solve a equation $\partial\mathcal{H}/\partial\mathbf{a} = \mathbf{0}$. So, it follows that

$$\mathbf{\Sigma}\mathbf{a} = \lambda\mathbf{a}. \tag{4.172}$$

Generally, λ is called an eigenvalue and $\mathbf{a}$ is an eigenvector.

The matrix $\mathbf{\Sigma}$ can be computed from observed data such that

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}[\mathbf{W} - \mathbf{e}\mathbf{m}^T]^T[\mathbf{W} - \mathbf{e}\mathbf{m}^T], \tag{4.173}$$

[58] Because $\zeta = \mathbf{a}^T\mathbf{x}$, $V(\zeta) = E[(\mathbf{a}^T\mathbf{x} - E(\mathbf{a}^T\mathbf{x}))(\mathbf{a}^T\mathbf{x} - E(\mathbf{a}^T\mathbf{x}))^T] = E[(\mathbf{a}^T(\mathbf{x} - E(\mathbf{x}))(\mathbf{a}^T(\mathbf{x} - E(\mathbf{x}))^T] = E[\mathbf{a}^T(\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^T\mathbf{a}] = \mathbf{a}^T E[(\mathbf{x} - E(\mathbf{x}))(\mathbf{x} - E(\mathbf{x}))^T]\mathbf{a} = \mathbf{a}^T\mathbf{\Sigma}\mathbf{a}$. Note that $E(\mathbf{a}^T\mathbf{x}) = \mathbf{a}^T E(\mathbf{x})$ since $\mathbf{a}$ is a weight vector independent of $\mathbf{x}$.

where $\mathbf{m}$ is an M-dimensional 'mean vector', j-th element of which is $m_j = N^{-1}\sum_{i=1}^{N} w_{ij}$ $(j = 1, ..., M)$ and $\mathbf{e}$ is an N-dimensional vector whose all elements are one (i.e., $\mathbf{e} = [1, 1, 1, ..., 1]^T$) [59].

By solving $\hat{\mathbf{\Sigma}}\mathbf{a} = \lambda\mathbf{a}$, weights for the linear combination of Equation (4.169) can be obtained. If the SVD of $\tilde{\mathbf{W}} = N^{-1/2}[\mathbf{W} - \mathbf{e}\mathbf{m}^T]$ is written as

$$\tilde{\mathbf{W}} = \frac{1}{\sqrt{N}}[\mathbf{W} - \mathbf{e}\mathbf{m}^T] = \mathbf{U}\mathbf{Q}\mathbf{V}^T, \tag{4.174}$$

then Equation (4.173) becomes

$$\hat{\mathbf{\Sigma}} = [\mathbf{U}\mathbf{Q}\mathbf{V}^T]^T\mathbf{U}\mathbf{Q}\mathbf{V}^T = \mathbf{V}\mathbf{Q}\mathbf{U}^T\mathbf{U}\mathbf{Q}\mathbf{V}^T = \mathbf{Q}\mathbf{Q}\mathbf{V}\mathbf{V}^T, \tag{4.175}$$

which leads to $\hat{\mathbf{\Sigma}}\mathbf{V} = \mathbf{Q}\mathbf{Q}\mathbf{V}$. In this context, it is possible to interpret that $\mathbf{V} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_r]$, and the corresponding element of diagonal matrix $\mathbf{Q}^2$ is an eigenvalue of eigenvector $\mathbf{a}_k$ $(k = 1, \ldots, r)$.

It is assumed that the diagonal elements of $\mathbf{Q}$ is arranged in descending order of their values like Equation (4.162), and that columns of $\mathbf{U}$ and $\mathbf{V}$ are also replaced in the same order here. Conventionally, variable ζ computed by eigenvector $\mathbf{a}_1$ is called 'first component', and that by $\mathbf{a}_2$ is 'second component' and so on. Figure 4.18 is a plot of documents in the sample DB by scores of the first component $\tilde{\mathbf{W}}\mathbf{a}_1$ (x-axis) and those of the second component $\tilde{\mathbf{W}}\mathbf{a}_2$ (y-axis). Actually, in the case of the sample DB,

$$\mathbf{V} = \begin{bmatrix} -0.286 & 0.112 & -0.212 & 0.707 & -0.217 & 0.561 \\ -0.670 & 0.210 & 0.245 & -0.535 & 0.043 & 0.400 \\ 0.105 & -0.754 & 0.551 & 0.039 & -0.154 & 0.303 \\ 0.271 & -0.224 & -0.451 & -0.205 & 0.613 & 0.507 \\ 0.608 & 0.546 & 0.376 & -0.090 & -0.152 & 0.398 \\ 0.124 & -0.164 & -0.497 & -0.404 & -0.727 & 0.136 \end{bmatrix}, \tag{4.176}$$

and for example, the first element of $\tilde{\mathbf{W}}\mathbf{a}_1$ is -0.667, which is a value for d_1. According to $\tilde{\mathbf{W}}\mathbf{V}$ obtained from the PCA, a graphical representation of document groups like this figure becomes feasible.

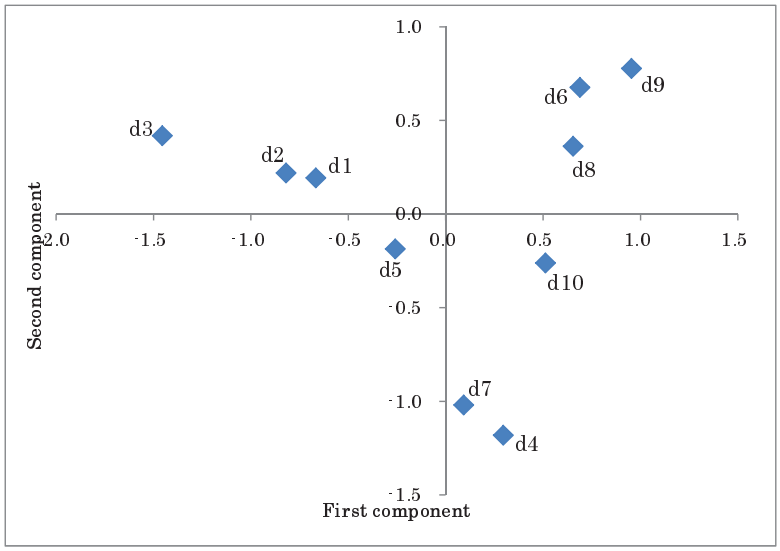


Figure 4.18: Result of PCA

As mentioned above, $V(\zeta) = \mathbf{a}^T\mathbf{\Sigma}\mathbf{a}$ and $\hat{\mathbf{\Sigma}}\mathbf{V} = \mathbf{Q}\mathbf{Q}\mathbf{V}$. Thus since $\mathbf{V}^T\hat{\mathbf{\Sigma}}\mathbf{V} = \mathbf{V}^T\mathbf{Q}\mathbf{Q}\mathbf{V} = \mathbf{Q}\mathbf{Q}\mathbf{V}^T\mathbf{V}$, variance of k-th component can be computed as $\eta_k^2\mathbf{a}_k^T\mathbf{a}_k = \eta_k^2$ where η_k means k-th element of $\mathbf{Q}$ as before. In the sample DB,

$$\mathbf{Q} = \mathrm{diag}\left[\ 2.339,\quad 1.998,\quad 1.623,\quad 1.226,\quad 0.825,\quad 0.372\ \right]. \tag{4.177}$$

[59] In statistical inference from a sample, $N - 1$ is often used instead of N in Equation (4.173).

So, the variance of first component is $2.339^2 = 5.472$, which amounts to about 37.9% of total variance of all principal components (i.e., $= 2.339^2/(2.339^2 + \ldots + 0.372^2)$). Because variance of the second component is about 27.7%, cumulative portion of variance up to the second component is computed as 65.6%, which can be considered to indicate the degree to which the plot in Figure 4.18 reflects variation in original data $\mathbf{W}$.

Otherwise, $\mathbf{U}$ in Equation (4.174) can be used for DC. For the sample DB, this matrix is computed such that

$$\mathbf{U} = \begin{bmatrix} -0.285 & 0.097 & -0.278 & 0.800 & -0.197 & 0.190 \\ -0.350 & 0.110 & -0.059 & -0.023 & 0.086 & -0.561 \\ -0.622 & 0.209 & 0.084 & -0.437 & 0.136 & 0.460 \\ 0.127 & -0.591 & 0.005 & 0.090 & 0.587 & 0.264 \\ -0.110 & -0.093 & 0.223 & -0.071 & -0.024 & -0.525 \\ 0.295 & 0.339 & 0.361 & -0.036 & -0.215 & 0.232 \\ 0.039 & -0.510 & 0.400 & -0.007 & -0.497 & 0.024 \\ 0.280 & 0.181 & -0.116 & 0.033 & 0.413 & -0.182 \\ 0.408 & 0.390 & 0.118 & 0.039 & 0.062 & 0.065 \\ 0.219 & -0.131 & -0.738 & -0.390 & -0.350 & 0.034 \end{bmatrix}, \tag{4.178}$$

whose the first column splits the dataset into $\{d_1, d_2, d_3, d_5\}$ and $\{d_4, d_6, d_7, d_8, d_9, d_{10}\}$ by setting zero as a border. Similarly, by using the second column, the former is furthermore partitioned into $\{d_1, d_2, d_3\}$ and $\{d_5\}$, and the latter becomes $\{d_4, d_7, d_{10}\}$ and $\{d_6, d_8, d_9\}$. Based on this heuristic rule, it is possible to obtain document clusters from a result of SVD after converting $\mathbf{W}$ as shown in Equation (4.174). According to this heuristic rule, documents are divided into four quadrants by the first and second components as indicated in Figure 4.18 (however, in the case of sample DB, it appears that d_5 and d_{10} are located in the middle area between three clusters) [60].

Needless to say, similar partitioning is feasible by using $\tilde{\mathbf{W}}\mathbf{V}$. It should be noted that, rather than using SVD of $\tilde{\mathbf{W}}$, the PCA can be executed by applying directly an algorithm for computing eigenvalues and eigenvectors of $\hat{\mathbf{\Sigma}}$ such as the Householder method, QR algorithm, Lanczos method and so on. In this case, document clustering is executed inevitably based on $\mathbf{V}$ because $\mathbf{U}$ is not explicitly computed.

In *principal direction divisive partitioning* (PDDP) algorithm [31, 32], which is a kind of binary divisive clustering (Section 3.2.4), the target document collection is repeatedly divided by using PCA. In each step, documents involved in a set are classified into two parts depending on whether the first component score is positive or negative. Also, in NGPDDP(non-greedy version of PDDP) algorithm [224], components other than the first one can be selected for the partition according to a criterion on the variance of a set of clusters. Meanwhile, PDDP(l) algorithm [331] tries to classify the target set into 2^l parts in each stage where $l \geq 1$ (i.e., if $l = 1$, then the PDDP(l) algorithm reduces to the original one) [61].

Another extension of the PDDP algorithm is to use *kernel PCA* [332], which is a nonlinear version of PCA. Let Φ denote a nonlinear map, $\Phi : \mathbb{R}^M \rightarrow F$ where $\mathbb{R}^M$ means an M-dimensional space of real numbers and F represents a feature space with arbitrarily high dimensionality [62]. When the target dataset can not be well classified into homogeneous subsets by any 'linear' function, the nonlinear map may enhance performance of the classification. Usually, based on the nonlinear map, a *kernel function* is defined as $k(\mathbf{d}_i, \mathbf{d}_h) \equiv \Phi(\mathbf{d}_i)^T \Phi(\mathbf{d}_h)$, which allows to not use directly a complicated nonlinear map. For example, Gaussian kernel is defined as $k(\mathbf{d}_i, \mathbf{d}_h) = \exp(-\|\mathbf{d}_i - \mathbf{d}_h\|^2/\sigma^2)$ where σ is a parameter [69] [63].

An $N \times N$ symmetric matrix whose (i, h)-element is $k(\mathbf{d}_i, \mathbf{d}_h)$ is denoted by $\mathbf{K}$ here, which can be decomposed as $\mathbf{K} = \mathbf{U}\mathbf{Q}\mathbf{U}^T$ based on the spectral decomposition theorem if $\mathbf{K}$ is a nonnegative

[60] Otherwise, like spectral clustering (Section 4.3.2), it may be possible to execute k-means clustering for some columns (e.g., the first and second ones) extracted from matrix $\mathbf{U}$.

[61] Only in this part, l does not denote the collection length.

[62] If the number of dimensions in F is lower than M, then the map can be considered as a dimensional reduction.

[63] Another example is polynomial kernel, $k(\mathbf{d}_i, \mathbf{d}_h) = (\mathbf{d}_i^T \mathbf{d}_h + c)^p$, where p is polynomial degree and c indicates a parameter [69]. If $p = 1$ and $c = 0$, then the kernel reduces to an inner product. 'Kernel-based clustering' techniques including support vector clustering (SVC) algorithm (see [322]) are useful for some problems (e.g., identification of handwritten digit patterns).

definite matrix [120]. Since

$$k(\mathbf{d}_i, \mathbf{d}_h) = [\mathbf{K}]_{ih} = [\mathbf{U}\mathbf{Q}\mathbf{U}^T]_{ih} = \sum_{j=1}^{r} \lambda_j u_{ij} u_{hj}, \tag{4.179}$$

where r is the rank of $\mathbf{K}$ (i.e., $r \leq N$) [64], λ_j is a diagonal element of $\mathbf{Q}$, and u_{ij} is (i,j)-element of $\mathbf{U}$, the nonlinear map can be represented as

$$\Phi(\mathbf{d}_i) = \left[\sqrt{\lambda_1} u_{i1}, \ldots, \sqrt{\lambda_r} u_{ir}\right]^T \tag{4.180}$$

($\Phi(\mathbf{d}_h)$ is similarly written) [69].

If $N^{-1}\sum_i \Phi(\mathbf{d}_i) = \mathbf{0}$ (i.e., the data is 'zero-centered'), then the covariance matrix can be written as

$$\hat{\mathbf{\Sigma}} = \frac{1}{N}\sum_{i=1}^{N} \Phi(\mathbf{d}_i)\Phi(\mathbf{d}_i)^T = \mathbf{\Phi}\mathbf{\Phi}^T, \tag{4.181}$$

where $\mathbf{\Phi} = [\Phi(\mathbf{d}_1)/\sqrt{N}, \ldots, \Phi(\mathbf{d}_N)/\sqrt{N}]$ (i.e., an $r \times N$ matrix). By substituting it into Equation (4.172), a kernel PCA becomes feasible. In the kernel PCA, it is assumed that eigenvector $\mathbf{a}$ in Equation (4.172) is a vector on the feature space, which means that the eigenvector is represented as $\mathbf{a} = \mathbf{\Phi}\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_N]^T$, or equivalently, $\mathbf{a} = \sum_i \alpha_i \Phi(\mathbf{d}_i)$ (α_i can be interpreted as a coefficient). Thus Equation (4.172) becomes that $\hat{\mathbf{\Sigma}}\mathbf{\Phi}\boldsymbol{\alpha} = \lambda\mathbf{\Phi}\boldsymbol{\alpha}$, which leads to $\mathbf{\Phi}^T\mathbf{\Phi}\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}$. Since $\mathbf{K} = N^{-1}\mathbf{\Phi}^T\mathbf{\Phi}$, finally,

$$\mathbf{K}\boldsymbol{\alpha} = N\lambda\boldsymbol{\alpha}, \tag{4.182}$$

is obtained. Thus by computing the eigenvalue decomposition under the constraint that $\lambda\boldsymbol{\alpha}^T\boldsymbol{\alpha} = 1$ for keeping $\mathbf{a}^T\mathbf{a} = 1$ [65], a result of the kernel PCA can be obtained [66]. A variant of PDDP(l) algorithm using the kernel PCA is called KPDDP(l) [332] [67].

4.2.3 Rescaling techniques for identifying minor clusters

Iterative residual rescaling (IRR) algorithm

An original intention of LSI is to identify major factors explaining variation of term occurrences among documents so that a minor portion causing 'noisy' search results is removed. However, in the case of DC, it may be useful to find clusters representing such minor portion. Iterative residual rescaling (IRR) algorithm [10, 11] tries to extract iteratively 'base vectors' corresponding to column vectors of $\mathbf{V}$ in Equation (4.158) with rescaling of the document-by-term matrix from which the base vectors are extracted. The residual matrix from which a vector is extracted in each iteration is denoted by $\mathbf{R}$ (an $N \times M$ matrix), and rescaling of the matrix is defined such that

$$\mathbf{R}_s = [\|\tilde{\mathbf{r}}_1\|^q\tilde{\mathbf{r}}_1, \|\tilde{\mathbf{r}}_2\|^q\tilde{\mathbf{r}}_2, ..., \|\tilde{\mathbf{r}}_N\|^q\tilde{\mathbf{r}}_N]^T, \tag{4.183}$$

where $\tilde{\mathbf{r}}_i$ indicates a column vector of $\mathbf{R}^T$ and q is a constant for rescaling ($q > 0$). Note that since $\tilde{\mathbf{r}}_i$ corresponds to a document vector, its norm can be considered as document length. Thus Equation (4.183) means that each document vector is augmented according to its length and that a longer document vector becomes further longer in the iterative steps.

If SVD of $\mathbf{R}$ is written as $\mathbf{R} = \mathbf{U}\mathbf{Q}\mathbf{V}^T$, then

$$\mathbf{R}^T\mathbf{R}\mathbf{V} = \mathbf{V}\mathbf{Q}^2, \tag{4.184}$$

as before (see Equation (4.175)), which means that each vector of $\mathbf{V}$ is an eigenvector of a symmetric matrix $\mathbf{R}^T\mathbf{R}$. Therefore, as an algorithm for obtaining $\mathbf{V}$ that is a right matrix of SVD, it is possible to compute Equation (4.184) repeatedly as follows:

[64] $[\cdot]_{ij}$ means (i,j)-element of the matrix.

[65] Note that $\mathbf{a}^T\mathbf{a} = \boldsymbol{\alpha}^T\mathbf{\Phi}^T\mathbf{\Phi}\boldsymbol{\alpha} = N^{-1}\boldsymbol{\alpha}^T\mathbf{K}\boldsymbol{\alpha} = N^{-1}\boldsymbol{\alpha}^T N\lambda\boldsymbol{\alpha} = \lambda\boldsymbol{\alpha}^T\boldsymbol{\alpha}$.

[66] When the dataset is not centered, matrix $\mathbf{K}$ has to be converted before computing the decomposition (see [322, 332]).

[67] As an application of DC, the kernel has been also used in 'semantic locally adaptive clustering' (semantic LAC) by [7].

1. Set that $k \leftarrow 1$ and record the rank of initial $\mathbf{R}$ into r.
2. Solve Equation (4.184) and keep its first eigenvector (i.e., the vector corresponding to the largest eigenvalue) as $\mathbf{v}_k$.
3. Compute projection [68] of $\mathbf{R}$ on a subspace constructed by $\mathbf{v}_k$ such that $\mathbf{R}\mathbf{v}_k\mathbf{v}_k^T$.
4. Compute residual portion by subtracting the projection from $\mathbf{R}$, and replace $\mathbf{R}$ by $\mathbf{R} - \mathbf{R}\mathbf{v}_k\mathbf{v}_k^T$, namely, $\mathbf{R} \leftarrow \mathbf{R} - \mathbf{R}\mathbf{v}_k\mathbf{v}_k^T$.
5. If $k = r$, then the procedure is terminated. Otherwise, return to step 2 after $k \leftarrow k + 1$.

By initially setting that $\mathbf{R} = \mathbf{W}$, this procedure allows to compute the right matrix in SVD of $\mathbf{W}$ (note that this is a part of general algorithm for calculating SVD).

In the IRR algorithm, before computing eigenvectors of $\mathbf{R}$, $\mathbf{R}$ is rescaled by Equation (4.183). If vectors of documents belonging to minor portion are enough enlarged by the rescaling, the IRR algorithm may find clearly it. The precise procedure of IRR algorithm is shown in Figure 4.19. The base vectors $\mathbf{b}_j$ $(j = 1. \ldots, m)$ correspond to components of right matrix of SVD, and if $q = 0$ and $m = M$, then it becomes that $\mathbf{b}_j = \mathbf{v}_j$ $(j = 1. \ldots, M)$, which means that the IRR algorithm reduces to computation of standard SVD.

‡ IRR algorithm

Set: The number of iterations m $(< M)$ and a constant q (> 0). Initially, $\mathbf{R} = \mathbf{W}$, and $k \leftarrow 1$.

1) Compute $\mathbf{R}_s$ by Equation (4.183) .

2) Compute eigenvectors of $\mathbf{R}_s^T\mathbf{R}_s$, and record the first eigenvector as $\mathbf{b}_k$.

3) Compute residual matrix as $\mathbf{R} \leftarrow \mathbf{R} - \mathbf{R}\mathbf{b}_k\mathbf{b}_k^T$.

4) $k \leftarrow k + 1$. If $k > m$, then the procedure is terminated. Otherwise, return to 1).

Out: Basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$.

Figure 4.19: IRR algorithm

After calculating the base vectors by the IRR algorithm, each document can be converted as

$$\bar{\mathbf{d}}_i = [\mathbf{b}_1, ..., \mathbf{b}_m]^T\mathbf{d}_i, \ i = 1, \ldots, N, \tag{4.185}$$

in which $\mathbf{b}_k$ is used for $\mathbf{v}_k$ of Equation (4.166) $(k = 1, \ldots, m)$. For example, the sample DB is converted by the IRR algorithm with $q = 2.0$ and $m = 3$ such that

$$\bar{\tilde{\mathbf{W}}}^T = \begin{bmatrix} 0.461 & 0.807 & 1.593 & -0.622 & 0.228 & -0.348 \\ 0.065 & 0.129 & 0.183 & -0.173 & -0.276 & -0.498 \\ -0.719 & -0.078 & 0.205 & 0.915 & 0.287 & -0.486 \end{bmatrix}$$

$$\begin{bmatrix} -0.335 & -0.511 & -0.619 & -0.654 \\ -0.522 & 0.054 & -0.243 & 1.281 \\ 0.865 & -0.376 & -0.701 & 0.090 \end{bmatrix},$$

where $\tilde{\mathbf{W}} = N^{-1/2}[\mathbf{W} - \mathbf{e}\mathbf{m}^T]$ as before and $\bar{\tilde{\mathbf{W}}}$ is a conversion of $\tilde{\mathbf{W}}$ according to Equation (4.185) (i.e., initially, $\mathbf{R} = \tilde{\mathbf{W}}$ in the IRR algorithm). Figure 4.20 is a plot of documents using the first and second rows of $\bar{\tilde{\mathbf{W}}}^T$, which provides a partition slightly different from that in Figure 4.18.

[68] In general, projection of a column vector $\mathbf{y}$ on a subspace representing a matrix $\mathbf{X}$ can be computed such that $\mathbf{z} = \mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$. Suppose that $\mathbf{z} = \mathbf{X}\mathbf{c}$ for vector $\mathbf{c}$ (which means that $\mathbf{z}$ is a vector on the subspace). If $\mathbf{z}$ is projection of $\mathbf{y}$ on $\mathbf{X}$, then $\mathbf{X}^T[\mathbf{y} - \mathbf{z}] = \mathbf{X}^T[\mathbf{y} - \mathbf{X}\mathbf{c}] = \mathbf{0}$ by its definition. Therefore, $\mathbf{c} = [\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$, which leads to $\mathbf{z} = \mathbf{X}[\mathbf{X}^T\mathbf{X}]^{-1}\mathbf{X}^T\mathbf{y}$. By replacing $\mathbf{X}$ with $\mathbf{v}_k$, it follows that $\mathbf{z} = \mathbf{v}_k[\mathbf{v}_k^T\mathbf{v}_k]^{-1}\mathbf{v}_k^T\mathbf{y}$. But, since $\mathbf{v}_k^T\mathbf{v}_k = 1$, it becomes that $\mathbf{z} = \mathbf{v}_k\mathbf{v}_k^T\mathbf{y}$ which leads to $\mathbf{z}^T = \mathbf{y}^T\mathbf{v}_k\mathbf{v}_k^T$ because $\mathbf{v}_k\mathbf{v}_k^T$ is a symmetric matrix. By replacing $\mathbf{y}$ with a document vector, projection $\mathbf{R}\mathbf{v}_k\mathbf{v}_k^T$ can be obtained.

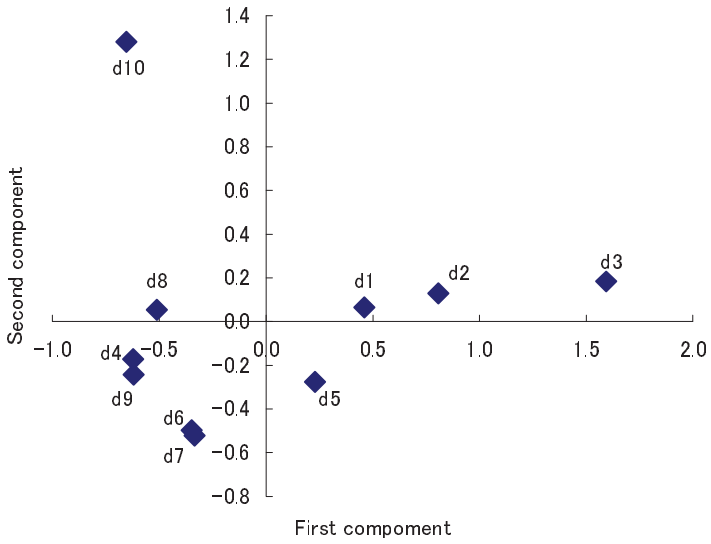


Figure 4.20: Result of IRR ($q = 2.0$)

As already mentioned, the scale factor q has an effect of augmenting length of each document vector so that longer vectors become further longer. For example, in the above case, the initial vector of d_1 is

$$[1.075, -0.126, -0.379, -0.284, -0.506, -0.158]^T,$$

which becomes

$$[1.804, -0.212, -0.637, -0.478, -0.849, -0.265]^T,$$

in the first stage. The rate of augmentation is rapidly increasing during the iteration. It may be possible to pick up outliers as minor clusters by using the augmentation of vectors.

COV-rescale algorithm

Unlike the IRR algorithm, COV-rescale algorithm [162] extracts base vectors from SVD of covariance (COV) matrix computed from rescaled residual matrix $\mathbf{R}_s$. So, without any rescaling operation, the COV-rescale algorithm computes standard SVD of a centralized matrix $\tilde{\mathbf{W}} = N^{-1/2}[\mathbf{W} - \mathbf{e}\mathbf{m}^T]$ (see Equation (4.173)) [69].

In the algorithm, the factor q is determined such that

$$q = \begin{cases} x^{-1} & \text{if } x > 1.0 \\ 1.0 + x & \text{if } x \approx 1.0 \\ 10^{1/x^2} & \text{if } x < 1.0 \end{cases}, \tag{4.186}$$

where $x = \max(\|\tilde{\mathbf{r}}_1\|, \ldots, \|\tilde{\mathbf{r}}_N\|)$. More specific procedure of the COV-rescale algorithm is shown as Figure 4.21.

For the sample DB, since $x = 6.08$ in the first step, it becomes that $q = 0.164$, which augments the first document vector $[4, 1, 0, 0, 0, 0]^T$ into

$$[5.049, 1.262, 0.000, 0.000, 0.000, 0.000]^T,$$

and finally, the algorithm generates a matrix

$$\bar{\mathbf{W}}^T = \begin{bmatrix} -1.786 & -2.332 & -4.390 & 1.285 & -0.534 & 2.404 \\ -0.561 & -0.659 & -1.224 & 3.713 & 0.593 & -2.182 \\ -0.394 & 0.526 & 1.207 & 0.868 & 1.955 & 2.721 \end{bmatrix}$$

[69] If $q = 0.0$, then the result from the IRR algorithm with $\tilde{\mathbf{W}}$ is equal to that from the COV-rescale algorithm with $\mathbf{W}$. But, if $q > 0.0$, then these results are not identical because the matrices rescaled in both the algorithms are different.

‡ COV-rescale algorithm

Set: The number of iterations m ($< M$). Initially, $\mathbf{R} = \mathbf{W}$, and $k \leftarrow 1$.

1) Compute q by Equation (4.186) after calculating norms of all vectors in $\mathbf{R}$.

2) Compute $\mathbf{R}_s$ by Equation (4.183).

3) Compute SVD of covariance matrix of $\mathbf{R}_s$, and record the first vector of its right matrix as $\mathbf{b}_k$.

4) Compute the residual matrix as $\mathbf{R} \leftarrow \mathbf{R} - \mathbf{R}\mathbf{b}_k\mathbf{b}_k^T$.

5) $k \leftarrow k + 1$. If $k > m$, then the procedure is terminated. Otherwise, return to 1).

Out: Basis vectors $\mathbf{b}_1, \ldots, \mathbf{b}_m$.

Figure 4.21: COV-rescale algorithm

$$\begin{bmatrix} 0.643 & 2.313 & 3.234 & 1.897 \\ 3.218 & -1.204 & -2.529 & 0.748 \\ 2.887 & 0.258 & 1.485 & -3.009 \end{bmatrix},$$

whose first and second rows provide a similar map with Figure 4.18 by the PCA.

4.2.4 Factorization into nonnegative matrices

Nonnegative matrix factorization

Nonnegative matrix factorization (NMF) has been recently used for identifying latent structure behind a phenomenon in various area including signal processing, neuroscience, data mining and so on [65], and also becomes being applied to DC problems [324]. According to the convention in literature, a term-by-document matrix $\mathbf{Z} \equiv \mathbf{W}^T = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_N]$ is considered to be an input to the NMF algorithm here. By the algorithm, original data set $\mathbf{Z}$ is approximately factorized into two non-negative matrices $\mathbf{A}$ and $\mathbf{C}$ such that $\mathbf{Z} \approx \mathbf{A}\mathbf{C}^T$ (all elements of the matrices are non-negative, i.e., $\mathbf{A} = [a_{ij}]$ where $a_{ij} \geq 0$ and $\mathbf{C} = [c_{ij}]$ where $c_{ij} \geq 0$) [70]. If $\mathbf{Z}$ is an $M \times N$ matrix, then $\mathbf{A}$ is an $M \times L$ matrix and $\mathbf{C}$ is an $N \times L$ matrix where $L < \min(N, M)$. It should be noted that $\mathbf{A}$ and $\mathbf{C}$ are not an orthogonal matrix, which is one of the differences from SVD [71].

In the case of DC, L corresponds to the number of clusters, which has to be determined a priori, and the matrix $\mathbf{C}$ obtained by numerical computation (see below) allows for allocating document d_i to cluster C_k such that

$$k = \underset{j=1,\ldots,L}{\arg\max}\, c_{ij}, \tag{4.187}$$

(i.e., $k = j$ where c_{ij} is the maximum among the set $\{c_{i1}, \ldots, c_{iL}\}$). Because $\mathbf{C}$ does not contain any negative element unlike $\mathbf{U}$ in the SVD, it is possible to allocate directly documents to clusters. This is an advantage of the NMF when it is employed for DC. In general, the non-negativity makes interpretation of results easier in connection with target entities in real world, which may be a reason that the NMF is widely applied in various areas. Note that terms in documents can be simultaneously grouped based on matrix $\mathbf{A}$ by a criterion similar to Equation (4.187).

For the NMF, it is necessary to find computationally $\mathbf{A}$ and $\mathbf{C}$ that minimize an objective function

$$\mathcal{H}(\mathbf{A}, \mathbf{C}) = \frac{1}{2}\|\mathbf{Z} - \mathbf{A}\mathbf{C}^T\|^2. \tag{4.188}$$

[70] Alternatively, the NMF can be formulated as $\mathbf{Z} = \mathbf{A}\mathbf{C}^T + \mathbf{E}$ where $\mathbf{E}$ denotes a matrix of error terms.

[71] When the orthogonality of $\mathbf{A}$ or $\mathbf{C}$ is assumed, then it is specially called 'semi-orthogonal NMF' [65]. In DC, it seems that the assumption of orthogonality is not adopted.

When using Frobenius norm [72] as $\|\cdot\|$, this objective function can be transformed such that

$$\begin{aligned}\mathcal{H}(\mathbf{A},\mathbf{C}) &= (1/2)\mathrm{tr}[(\mathbf{Z}-\mathbf{A}\mathbf{C}^T)(\mathbf{Z}-\mathbf{A}\mathbf{C}^T)^T] \\ &= (1/2)\mathrm{tr}(\mathbf{Z}\mathbf{Z}^T)-\mathrm{tr}(\mathbf{Z}\mathbf{C}\mathbf{A}^T)+(1/2)\mathrm{tr}(\mathbf{A}\mathbf{C}^T\mathbf{C}\mathbf{A}^T),\end{aligned} \tag{4.189}$$

according to its definition [73].

Since this is a minimization problem with inequality constrains that $a_{ij} \geq 0$ and $c_{ij} \geq 0$ for all i and j, it is necessary to apply Karush-Kuhn-Tucker (KKT) theorem for solving it. Suppose that $f(\mathbf{x})$ has to be minimized under a condition $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$ where $\mathbf{x}$ is an n-dimensional vector and $f : \mathbb{R}^n \to \mathbb{R}$ and $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^p$. According to the KKT theorem, there exists $\boldsymbol{\alpha} \in \mathbb{R}^p$ satisfying conditions such that

1. $\boldsymbol{\alpha} \geq \mathbf{0}$
2. $\mathcal{D}f(\mathbf{x}^*)^T - \mathcal{D}\mathbf{g}(\mathbf{x}^*)^T\boldsymbol{\alpha} = \mathbf{0}$
3. $\mathbf{g}(\mathbf{x}^*)^T\boldsymbol{\alpha} = 0$

where $\mathbf{x}^* = [x_1^*, \ldots, x_n^*]^T$ is a local minimizer and $\mathcal{D}$ means differentiation [74] [63]. The vector $\boldsymbol{\alpha}$ is called the KKT multiplier vector.

In the case of NMF, Equation (4.189) corresponds to the function $f(\mathbf{x}^*)$ in the conditions, and its derivatives can be computed such that

$$\frac{\partial\mathcal{H}(\mathbf{A},\mathbf{C})}{\partial\mathbf{A}} = -\mathbf{Z}\mathbf{C}+\mathbf{A}\mathbf{C}^T\mathbf{C}, \tag{4.190}$$

$$\frac{\partial\mathcal{H}(\mathbf{A},\mathbf{C})}{\partial\mathbf{C}} = -\mathbf{Z}^T\mathbf{A}+\mathbf{C}\mathbf{A}^T\mathbf{A}, \tag{4.191}$$

from the definition of trace [75]. Also, since $p = n$ and $\mathbf{g}(\mathbf{x}) = \mathbf{x}$ (i.e., $g_1(\mathbf{x}) = x_1, \ldots, g_n(\mathbf{x}) = x_n$) in the case of NMF, $\mathcal{D}\mathbf{g}(\mathbf{x}) = \mathbf{I}_{(n)}$ and $\mathcal{D}\mathbf{g}(\mathbf{x})^T\boldsymbol{\alpha} = \boldsymbol{\alpha}$ are obtained where $\mathbf{I}_{(n)}$ denotes an n-dimensional diagonal matrix whose all diagonal elements are one. Therefore, for computing NMF,

[72] Frobenius norm of a matrix $\mathbf{X} = [x_{ij}]$ is defined as

$$\|\mathbf{X}\| = \left(\sum_i\sum_j x_{ij}^2\right)^{1/2}.$$

[73] If $\mathbf{X}$ is a 2×2 matrix, then

$$\mathbf{X}\mathbf{X}^T = \begin{bmatrix} x_{11}^2+x_{12}^2 & x_{11}x_{21}+x_{12}x_{22} \\ x_{11}x_{21}+x_{12}x_{22} & x_{21}^2+x_{22}^2 \end{bmatrix},$$

and since the trace of a square matrix is defined as the sum of its diagonal elements,

$$\mathrm{tr}(\mathbf{X}\mathbf{X}^T) = \sum_{i=1}^{2}\sum_{j=1}^{2} x_{ij}^2 = \|\mathbf{X}\|^2.$$

Also, $\mathrm{tr}(\mathbf{X}\mathbf{Y}^T) = \mathrm{tr}(\mathbf{Y}^T\mathbf{X})$ where $\mathbf{X}$ and $\mathbf{Y}$ are an $n\times m$ matrix, and $\mathrm{tr}(\mathbf{X}-\mathbf{Y}) = \mathrm{tr}(\mathbf{X}) - \mathrm{tr}(\mathbf{Y})$ when both $\mathbf{X}$ and $\mathbf{Y}$ are a square matrix [120].

[74] Since $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_p(\mathbf{x})]^T$, for a point $\mathbf{x} = \mathbf{x}_0$,

$$\mathcal{D}\mathbf{g}(\mathbf{x}_0) = \left[\frac{\partial\mathbf{g}}{\partial x_1}(\mathbf{x}_0), \ldots, \frac{\partial\mathbf{g}}{\partial x_n}(\mathbf{x}_0)\right] = \begin{bmatrix} \frac{\partial g_1}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial g_1}{\partial x_n}(\mathbf{x}_0) \\ \vdots & \ddots & \vdots \\ \frac{\partial g_p}{\partial x_1}(\mathbf{x}_0) & \cdots & \frac{\partial g_p}{\partial x_n}(\mathbf{x}_0) \end{bmatrix},$$

which is called Jacobian matrix or derivative matrix. Also, $\mathcal{D}f(\mathbf{x}) = [\partial f(\mathbf{x})/\partial x_1, \ldots, \partial f(\mathbf{x})/\partial x_n]$ according to the definition.

[75] For an $m\times n$ matrix $\mathbf{X} = [x_{ij}]$ and an $n\times m$ matrix $\mathbf{Y} = [y_{ij}]$, $\partial\mathrm{tr}[\mathbf{Y}\mathbf{X}]/\partial x_{ij} = \partial\mathrm{tr}[\mathbf{X}\mathbf{Y}]/\partial x_{ij} = y_{ji}$. Therefore,

$$\frac{\partial\mathrm{tr}[\mathbf{Y}\mathbf{X}]}{\partial\mathbf{X}} = \frac{\partial\mathrm{tr}[\mathbf{X}\mathbf{Y}]}{\partial\mathbf{X}} = \mathbf{Y}^T,$$

and similarly, when $\mathbf{X}$ is an $n\times m$ matrix,

$$\frac{\partial\mathrm{tr}[\mathbf{Y}\mathbf{X}^T]}{\partial\mathbf{X}} = \frac{\partial\mathrm{tr}[\mathbf{X}^T\mathbf{Y}]}{\partial\mathbf{X}} = \mathbf{Y}.$$

the second KKT condition related to Equation (4.190) becomes

$$-\mathbf{ZC} + \mathbf{AC}^T\mathbf{C} - \bar{\mathbf{A}}_1 = \mathbf{O}, \tag{4.192}$$

where $\bar{\mathbf{A}}_1 = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_L]$, $\boldsymbol{\alpha}_j$ is an M-dimensional KKT multiplier vector ($j = 1, \ldots, L$), and $\mathbf{O}$ denotes a zero matrix. Similarly,

$$-\mathbf{Z}^T\mathbf{A} + \mathbf{CA}^T\mathbf{A} - \bar{\mathbf{A}}_2 = \mathbf{O}, \tag{4.193}$$

where $\bar{\mathbf{A}}_2 = [\tilde{\boldsymbol{\alpha}}_1, \ldots, \tilde{\boldsymbol{\alpha}}_L]$ and $\tilde{\boldsymbol{\alpha}}_j$ is an N-dimensional KKT multiplier vector.

Since the third KKT condition becomes $x^*_{j'} \times \alpha_{j'} = 0$ where $\alpha_{j'}$ indicates an element of $\boldsymbol{\alpha}$ ($j' = 1, \ldots, n$) in the case that $p = n$ and $g_1(\mathbf{x}) = x_1, \ldots, g_n(\mathbf{x}) = x_n$, the KKT multiplier vector can be eliminated such that

$$-\mathbf{ZC} \circ \mathbf{A} + \mathbf{AC}^T\mathbf{C} \circ \mathbf{A} - \bar{\mathbf{A}}_1 \circ \mathbf{A} = -\mathbf{ZC} \circ \mathbf{A} + \mathbf{AC}^T\mathbf{C} \circ \mathbf{A} = \mathbf{O}, \tag{4.194}$$

where $\circ$ means the Hadamard product [76]. Similarly,

$$-\mathbf{Z}^T\mathbf{A} \circ \mathbf{C} + \mathbf{CA}^T\mathbf{A} \circ \mathbf{C} = \mathbf{O}, \tag{4.195}$$

and Equations (4.194) and (4.195) lead to

$$-[\mathbf{ZC}]_{ij}a_{ij} + [\mathbf{AC}^T\mathbf{C}]_{ij}a_{ij} = 0, \tag{4.196}$$

$$-[\mathbf{Z}^T\mathbf{A}]_{ij}c_{ij} + [\mathbf{CA}^T\mathbf{A}]_{ij}c_{ij} = 0, \tag{4.197}$$

for each element of $\mathbf{A}$ and $\mathbf{C}$. From the equations, updating formulas for computing numerically $\mathbf{A}$ and $\mathbf{C}$ can be derived such that

$$a_{ij} \leftarrow a_{ij}\frac{[\mathbf{ZC}]_{ij}}{[\mathbf{AC}^T\mathbf{C}]_{ij}}, \tag{4.198}$$

$$c_{ij} \leftarrow c_{ij}\frac{[\mathbf{Z}^T\mathbf{A}]_{ij}}{[\mathbf{CA}^T\mathbf{A}]_{ij}}. \tag{4.199}$$

Therefore, by updating repeatedly each value of a_{ij} ($i = 1, \ldots, M$; $j = 1, \ldots, L$) and c_{ij} ($i = 1, \ldots, N$; $j = 1, \ldots, L$) according to Equations (4.198) and (4.199), respectively, after allocating randomly initial positive values to them, NMF can be executed. This computational method is often called *multiplicative iterative algorithm* or 'Lee-Seung algorithm' [180], which is relatively easier to be implemented than other estimation techniques such as Newton's method.

However, $\mathbf{A}$ and $\mathbf{C}$ obtained by the algorithm using Equations (4.198) and (4.199) are not unique because clearly

$$\mathbf{Z} \approx \mathbf{AC}^T = [\mathbf{AD}][\mathbf{CD}^{-1}]^T \tag{4.200}$$

for a diagonal matrix $\mathbf{D}$. In this section, each column of $\mathbf{A}$ is converted into a unit vector based on the norm, namely, elements on j-th column are divided by $(\sum_i a_{ij}^2)^{1/2}$ for obtaining a unique result.

By this algorithm of NMF under an assumption that $L = 3$, the sample DB can be approximately factorized such that

$$\mathbf{A} = \begin{bmatrix} 0.323 & 0.000 & 0.000 \\ 0.945 & 0.024 & 0.010 \\ 0.036 & 0.907 & 0.000 \\ 0.000 & 0.358 & 0.211 \\ 0.019 & 0.040 & 0.974 \\ 0.000 & 0.216 & 0.081 \end{bmatrix},$$

[76] If $\mathbf{X}$ and $\mathbf{Y}$ are a 2×2 matrix, then

$$\mathbf{X} \circ \mathbf{Y} = \begin{bmatrix} x_{11} \times y_{11} & x_{12} \times y_{12} \\ x_{21} \times y_{21} & x_{22} \times y_{22} \end{bmatrix},$$

which is element-by-element multiplication of two matrices with the same dimension.

and

$$\mathbf{C}^T = \begin{bmatrix} 2.238 & 3.160 & 5.996 & 0.000 & 1.867 & 0.916 & 0.870 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 4.701 & 1.702 & 0.446 & 4.711 & 0.401 & 0.000 & 1.754 \\ 0.000 & 0.000 & 0.000 & 0.013 & 0.718 & 4.796 & 0.420 & 3.291 & 5.081 & 1.700 \end{bmatrix}.$$

Because

$$[\mathbf{AC}^T]^T = \begin{bmatrix} 0.724 & 2.116 & 0.081 & 0.000 & 0.043 & 0.000 \\ 1.021 & 2.987 & 0.114 & 0.000 & 0.060 & 0.000 \\ 1.938 & 5.669 & 0.217 & 0.000 & 0.115 & 0.000 \\ 0.000 & 0.111 & 4.265 & 1.685 & 0.199 & 1.017 \\ 0.603 & 1.813 & 1.612 & 0.761 & 0.802 & 0.426 \\ 0.296 & 0.926 & 0.437 & 1.170 & 4.707 & 0.485 \\ 0.281 & 0.938 & 4.306 & 1.775 & 0.612 & 1.052 \\ 0.000 & 0.044 & 0.364 & 0.837 & 3.222 & 0.353 \\ 0.000 & 0.053 & 0.000 & 1.071 & 4.950 & 0.412 \\ 0.000 & 0.059 & 1.591 & 0.986 & 1.725 & 0.517 \end{bmatrix}, \tag{4.201}$$

it is clear that $\mathbf{AC}^T$ is not exactly equivalent to $\mathbf{Z}$ and just an approximation of it.

The matrix $\mathbf{C}^T$ indicates that the sample DB was partitioned into $\{d_1, d_2, d_3, d_5\}$ (with the first row), $\{d_4, d_7, d_{10}\}$ (with the second row), and $\{d_6, d_8, d_9\}$ (with the third row), which is almost consistent with the results from other techniques. Concurrently, by estimated $\mathbf{A}$, terms can be grouped into $\{t_1, t_2\}$ (with the first column), $\{t_3, t_4, t_6\}$ (with the second column) and $\{t_5\}$ (with the third column). Alternatively, in the case of using unit vectors as input data (i.e., $\mathbf{Z} = [\tilde{\mathbf{d}}_1, \ldots, \tilde{\mathbf{d}}_N]$), the set of clusters became $\{d_1, d_2, d_3\}$, $\{d_4, d_5, d_7\}$ and $\{d_6, d_8, d_9, d_{10}\}$ with term clusters $\{t_1, t_2\}$, $\{t_3, t_6\}$ and $\{t_4, t_5\}$.

More specifically, the above matrices were computed based on Equations (4.198) and (4.199) as shown in Figure 4.22. Actually, a package of R for NMF becomes available [103, 102], and it is not necessary to write source code for implementing NMF if using the package.

‡ Clustering by NMF

Set: The number of clusters L.

1) Set initial $\mathbf{A}$ and $\mathbf{C}$ by allocating a random number to each element.

2) Iterate the following process from a) to d). If $\mathbf{A}$ and $\mathbf{C}$ converge or the times of iterations exceed a prescribed number, then go to step 3).

   a) Compute $x_{ij} = [\mathbf{ZC}]_{ij}/[\mathbf{AC}^T\mathbf{C}]_{ij}$ $(i = 1, \ldots, M; j = 1, \ldots, L)$.

   b) Compute $y_{ij} = [\mathbf{Z}^T\mathbf{A}]_{ij}/[\mathbf{CA}^T\mathbf{A}]_{ij}$ $(i = 1, \ldots, N; j = 1, \ldots, L)$.

   c) Update $\mathbf{A}$ and $\mathbf{C}$ by $a_{ij} \leftarrow a_{ij}x_{ij}$ and $c_{ij} \leftarrow c_{ij}y_{ij}$ for all elements.

   d) Divide each element of j-th column of $\mathbf{A}$ by $(\sum_i a_{ij}^2)^{1/2}$.

3) Generate clusters based on Equation (4.187) and terminate the procedure.

Out: Cluster set $C_1, \ldots, C_L$, and $\mathbf{A}$ and $\mathbf{C}$.

Figure 4.22: Clustering by NMF

One of the problems in computing NMF is instability of results. Like other iterative algorithms, a series of iterative computation may fall into a local minimum. Also, it is not guaranteed that different algorithms for estimating $\mathbf{A}$ and $\mathbf{C}$ reach always to the same approximation of them.

Nonnegative block value decomposition

In *nonnegative block value decomposition* (NBVD) [196], a 'block' matrix $\mathbf{B}$ is additionally incorporated into the middle position of NMF such that

$$\mathbf{Z} \approx \mathbf{ABC}^T, \tag{4.202}$$

where $\mathbf{A}$ ($M \times L'$ matrix), $\mathbf{B}$ ($L' \times L$ matrix) and $\mathbf{C}$ ($N \times L$ matrix) are non-negative matrices, where $L' < M$ and $L < N$ [77]. If $\mathbf{B} = \mathbf{I}_{(L)}$ and $\mathbf{A}$ is an $N \times L$ matrix (i.e., $L' = L$), then the NBVD in Equation (4.202) reduces to NMF. Similarly, the objective function in estimation of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ corresponding to Equation (4.188) becomes

$$\mathcal{H}(\mathbf{A}, \mathbf{B}, \mathbf{C}) = \frac{1}{2}\|\mathbf{Z} - \mathbf{A}\mathbf{B}\mathbf{C}^T\|^2. \tag{4.203}$$

Actually, NBVD of the sample DB can be estimated such that

$$\mathbf{A} = \begin{bmatrix} 0.244 & 0.000 & 0.000 \\ 0.714 & 0.015 & 0.008 \\ 0.027 & 0.599 & 0.000 \\ 0.000 & 0.233 & 0.165 \\ 0.014 & 0.011 & 0.763 \\ 0.000 & 0.141 & 0.063 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0.000 & 19.922 & 0.000 \\ 0.000 & 0.000 & 20.770 \\ 20.396 & 0.000 & 0.461 \end{bmatrix},$$

and

$$\mathbf{C}^T = \begin{bmatrix} 0.000 & 0.000 & 0.000 & 0.000 & 0.045 & 0.300 & 0.025 & 0.206 & 0.318 & 0.106 \\ 0.149 & 0.210 & 0.398 & 0.000 & 0.124 & 0.061 & 0.058 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.343 & 0.124 & 0.032 & 0.344 & 0.029 & 0.000 & 0.128 \end{bmatrix}.$$

The matirx $\mathbf{C}^T$ suggests that documents are partitioned into $\{d_6, d_8, d_9\}$ (with the first row), $\{d_1, d_2, d_3\}$ (with the second row), and $\{d_4, d_5, d_7, d_{10}\}$ (with the third row) (where more precisely, $c_{52} = 0.12406$ and $c_{53} = 0.12412$). Also, in this case, $[\mathbf{A}\mathbf{B}\mathbf{C}^T]^T$ becomes almost equivalent to Equation (4.201) obtained by NMF.

Because $\mathbf{A}$ and $\mathbf{C}$ are normalized in this estimation so that the sum of elements in each column becomes one (see below), each element of matrix $\mathbf{B}$ indicates the times that terms belonging to i-th term cluster occur in documents included in j-th document cluster. Note that the sum of all elements in $\mathbf{B}$ amounts to about 61.55, which equals almost to the total number of tokens in the sample DB (i.e., = 62).

For this estimation of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$, a method almost similar to that from Equation (4.190) to (4.199) can be used. Since Equation (4.203) becomes

$$\begin{aligned} \mathcal{H}(\mathbf{A}, \mathbf{B}, \mathbf{C}) &= \frac{1}{2}\mathrm{tr}[(\mathbf{Z} - \mathbf{A}\mathbf{B}\mathbf{C}^T)(\mathbf{Z} - \mathbf{A}\mathbf{B}\mathbf{C}^T)^T] \\ &= \frac{1}{2}\mathrm{tr}(\mathbf{Z}\mathbf{Z}^T) - \mathrm{tr}(\mathbf{Z}\mathbf{C}\mathbf{B}^T\mathbf{A}^T) + \frac{1}{2}\mathrm{tr}(\mathbf{A}\mathbf{B}\mathbf{C}^T\mathbf{C}\mathbf{B}^T\mathbf{A}^T), \end{aligned} \tag{4.204}$$

by calculating $\partial\mathcal{H}/\partial\mathbf{A}$, $\partial\mathcal{H}/\partial\mathbf{B}$ and $\partial\mathcal{H}/\partial\mathbf{C}$ and considering the KKT second condition, it follows that

$$\mathbf{Z}\mathbf{C}\mathbf{B}^T - \mathbf{A}\mathbf{B}\mathbf{C}^T\mathbf{C}\mathbf{B}^T + \bar{\mathbf{B}}_1 = \mathbf{O}, \tag{4.205}$$

$$\mathbf{A}^T\mathbf{Z}\mathbf{C} - \mathbf{A}^T\mathbf{A}\mathbf{B}\mathbf{C}^T\mathbf{C} + \bar{\mathbf{B}}_2 = \mathbf{O}, \tag{4.206}$$

$$\mathbf{Z}^T\mathbf{A}\mathbf{B} - \mathbf{C}\mathbf{B}^T\mathbf{A}^T\mathbf{A}\mathbf{B} + \bar{\mathbf{B}}_3 = \mathbf{O}, \tag{4.207}$$

where $\bar{\mathbf{B}}_h$ is a matrix consisting of KKT multiplier vectors ($h = 1, 2, 3$). Finally, by applying '$\circ\mathbf{A}$', '$\circ\mathbf{B}$' and '$\circ\mathbf{C}$' to both sides of these equations, respectively, so that $\bar{\mathbf{B}}_h$ is eliminated ($h = 1, 2, 3$) based on the third KKT conditions (i.e., $\bar{\mathbf{B}}_1 \circ \mathbf{A} = \mathbf{O}$, $\bar{\mathbf{B}}_2 \circ \mathbf{B} = \mathbf{O}$ and $\bar{\mathbf{B}}_3 \circ \mathbf{C} = \mathbf{O}$) as before, updating formulas for the NBVD become

$$[\mathbf{A}]_{ij} \leftarrow [\mathbf{A}]_{ij} \frac{[\mathbf{Z}\mathbf{C}\mathbf{B}^T]_{ij}}{[\mathbf{A}\mathbf{B}\mathbf{C}^T\mathbf{C}\mathbf{B}^T]_{ij}}, \tag{4.208}$$

$$[\mathbf{B}]_{ij} \leftarrow [\mathbf{B}]_{ij} \frac{[\mathbf{A}^T\mathbf{Z}\mathbf{C}]_{ij}}{[\mathbf{A}^T\mathbf{A}\mathbf{B}\mathbf{C}^T\mathbf{C}]_{ij}}, \tag{4.209}$$

[77] The decomposition in Equation (4.202) can be considered as a special case of 'three-factor NMF' or 'tri-NMF'. Actually, there are many versions of NMF with varying constraints on the matrix components (see [65]). Also, decomposition similar to NBVD has been proposed by [323], which calls it 'concept factorization'.

$$[\mathbf{C}]_{ij} \leftarrow [\mathbf{C}]_{ij} \frac{[\mathbf{Z}^T \mathbf{A}\mathbf{B}]_{ij}}{[\mathbf{C}\mathbf{B}^T \mathbf{A}^T \mathbf{A}\mathbf{B}]_{ij}}. \tag{4.210}$$

If Equations (4.208), (4.209) and (4.210) are used in iterative procedure like that for NMF after $\mathbf{B}$ is also initialized by allocating random numbers, NBVD can be similarly computed for a given data matrix $\mathbf{Z}$. However, in the NBVD, normalization in 2-d) step of the procedure is slightly different as mentioned above. Precisely, a_{ij} is replaced by $a_{ij}/\sum_{i'} a_{i'j}$ and c_{ij} by $c_{ij}/\sum_{i'} c_{i'j}$ at the end of each iteration, which means that the sum of elements in each column of $\mathbf{A}$ and $\mathbf{C}$ becomes one (normalization based on unit vectors would be feasible).

Nonnegative tensor factorization

Target data are assumed here to be represented as three-way array $\mathcal{X} = [x_{ijk}]$ $(i = 1, \ldots, n;$ $j = 1, \ldots, m;$ $k = 1, \ldots, p)$. For example, if terms appearing in a set of e-mails are counted by author and month, then x_{ijk} may indicate occurrence frequency of j-th term in e-mails written by i-th author on k-th month (i.e,. 'author × term × time' data).

One of the techniques for analyzing such complicated data is PARAFAC (parallel factors) model [116], which is widely applied in various areas. In this model, a data element x_{ijk} in $\mathcal{X}$ is assumed to be decomposed as

$$x_{ijk} = \sum_{h=1}^{L} a_{ih} b_{jh} c_{kh}, \quad i = 1, \ldots, n; j = 1, \ldots, m; k = 1, \ldots, p, \tag{4.211}$$

which resembles SVD or NMF in that each data element is explained as the sum of L components $a_{ih}b_{jh}c_{kh}$ $(h = 1, \ldots, L)$ (see Figure 4.23).

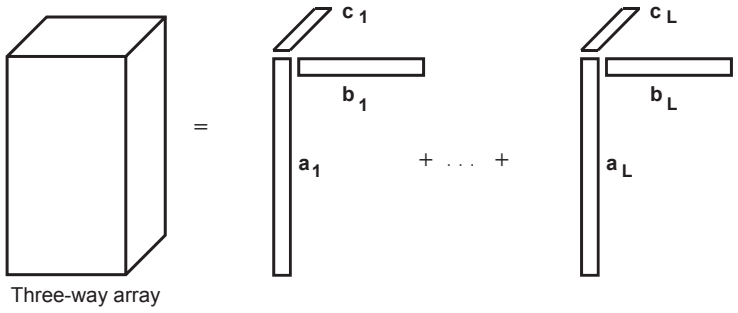


Figure 4.23: PARAFAC model

For estimating a_{ih}, b_{jh} and c_{kh}, it is necessary to convert Equation (4.211) into matrix form. This conversion is often called 'unfolding' or 'matricization'. For example, three-way array $\mathcal{X}$ can be represented as a matrix

$$\mathbf{X}^{(n \times mp)} = \begin{bmatrix} x_{111} & x_{121} & \cdots & x_{1m1} & x_{112} & \cdots & x_{1m2} & \cdots & x_{11p} & \cdots & x_{1mp} \\ x_{211} & x_{221} & \cdots & x_{2m1} & x_{212} & \cdots & x_{2m2} & \cdots & x_{21p} & \cdots & x_{2mp} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ x_{n11} & x_{n21} & \cdots & x_{nm1} & x_{n12} & \cdots & x_{nm2} & \cdots & x_{n1p} & \cdots & x_{nmp} \end{bmatrix}, \tag{4.212}$$

by putting x_{ijk} into (i, j')-element of the matrix where $j' = (k-1) \times m + j$ (i.e., index j runs faster than k).

Meanwhile, it is assumed that $\mathbf{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_L]$, $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_L]$, and $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_L]$, where $\boldsymbol{a}_h = [a_{1h}, \ldots, a_{nh}]^T$, $\mathbf{b}_h = [b_{1h}, \ldots, b_{mh}]^T$, and $\mathbf{c}_h = [c_{1h}, \ldots, c_{ph}]^T$, respectively $(h = 1, \ldots, L)$. Since the Khatri-Rao product can be defined as

$$\mathbf{C} * \mathbf{B} \equiv [\mathbf{c}_1 \otimes \mathbf{b}_1, \ldots, \mathbf{c}_L \otimes \mathbf{b}_L], \tag{4.213}$$

where $\otimes$ indicates the Kronecker product [78], it becomes that

$$[\mathbf{C} * \mathbf{B}]^T = \begin{bmatrix} b_{11}c_{11} & b_{21}c_{11} & \cdots & b_{m1}c_{11} & \cdots & b_{11}c_{p1} & \cdots & b_{m1}c_{p1} \\ b_{12}c_{12} & b_{22}c_{12} & \cdots & b_{m2}c_{12} & \cdots & b_{12}c_{p2} & \cdots & b_{m2}c_{p2} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{1L}c_{1L} & b_{2L}c_{1L} & \cdots & b_{mL}c_{1L} & \cdots & b_{1L}c_{pL} & \cdots & b_{mL}c_{pL} \end{bmatrix}. \tag{4.214}$$

So, (i, j')-element of an $n \times mp$ matrix $\mathbf{A}[\mathbf{C} * \mathbf{B}]^T$ can be written as

$$[\mathbf{A}[\mathbf{C} * \mathbf{B}]^T]_{ij'} = \sum_{h=1}^{L} a_{ih} b_{jh} c_{kh}, \tag{4.215}$$

where $j' = (k-1) \times m + j$ again.

By combining Equations (4.212) and (4.215), the PARAFAC model in Equation (4.211) can be represented as a product of two matrices such that

$$\mathbf{X}^{(n \times mp)} = \mathbf{A}[\mathbf{C} * \mathbf{B}]^T = \mathbf{A}\mathbf{Z}_1^T, \tag{4.216}$$

where $\mathbf{Z}_1 \equiv \mathbf{C} * \mathbf{B}$. Similarly, by changing the turn of running indexes, i, j and k, it follows that

$$\mathbf{X}^{(m \times pn)} = \mathbf{B}[\mathbf{A} * \mathbf{C}]^T = \mathbf{B}\mathbf{Z}_2^T, \tag{4.217}$$

$$\mathbf{X}^{(p \times nm)} = \mathbf{C}[\mathbf{B} * \mathbf{A}]^T = \mathbf{C}\mathbf{Z}_3^T, \tag{4.218}$$

where $\mathbf{Z}_2 \equiv \mathbf{A} * \mathbf{C}$ and $\mathbf{Z}_3 \equiv \mathbf{B} * \mathbf{A}$.

Although the PARAFAC model can be solved by alternating least squares (ALS) algorithms, the multiplicative iterative algorithm (Lee-Seung algorithm) can be applied if $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are assumed to be nonnegative [315]. Because Equation (4.216) can be considered as NMF of $\mathbf{X}^{(n \times mp)}$, by repeating the procedure for deriving Equation (4.198), an updating formula is obtained as

$$a_{ih'} \leftarrow a_{ih'} \frac{[\mathbf{X}^{(n \times mp)}\mathbf{Z}_1]_{ih'}}{[\mathbf{A}\mathbf{Z}_1^T\mathbf{Z}_1]_{ih'}}, \ h' = 1, \ldots, mp, \tag{4.219}$$

and similarly, Equations (4.217) and (4.218) lead to

$$b_{jh'} \leftarrow b_{jh'} \frac{[\mathbf{X}^{(m \times pn)}\mathbf{Z}_2]_{jh'}}{[\mathbf{B}\mathbf{Z}_2^T\mathbf{Z}_2]_{jh'}}, \ h' = 1, \ldots, pn, \tag{4.220}$$

and

$$c_{kh'} \leftarrow c_{kh'} \frac{[\mathbf{X}^{(p \times nm)}\mathbf{Z}_3]_{kh'}}{[\mathbf{C}\mathbf{Z}_3^T\mathbf{Z}_3]_{kh'}}, \ h' = 1, \ldots, nm. \tag{4.221}$$

Therefore, since $\mathbf{Z}_1 \equiv \mathbf{C} * \mathbf{B}$, $\mathbf{Z}_2 \equiv \mathbf{A} * \mathbf{C}$ and $\mathbf{Z}_3 \equiv \mathbf{B} * \mathbf{A}$, by allocating random numbers to elements of $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ and updating them iteratively based on Equations (4.219), (4.220) and (4.221), the estimation may be feasible.

When non-negativity is imposed into matrices of PARAFAC, it is called 'nonnegative PARAFAC' or *nonnegative tensor factorization* (NTF) [79] [14, 65]. Technically, the NTF can be applied to a model including four or more factors (see [65] for details).

4.3 Graph-based Document Clustering

It is possible to execute document clustering (DC) based on graph theory. Since a graph can be represented by a matrix as described below, the graph-based clustering has a close relationship with the matrix-based clustering discussed in previous section.

[78] For instance, $\mathbf{c}_1 \otimes \mathbf{b}_1 = [c_{11}\mathbf{b}_1^T, c_{21}\mathbf{b}_1^T, \ldots, c_{p1}\mathbf{b}_1^T]^T$, which is an $(m \times p)$-dimensional vector.

[79] NTF is sometime called 'positive tensor factorization' (PTF) (e.g., see [315]).

4.3.1 Graph cut

Definition of graph

Formally, a *graph* (or network) is defined as a set $G = \{V, E\}$ where V is a collection of vertices (or nodes), which is denoted by $V = \{v_1, v_2, \ldots, v_N\}$, and E is a collection of edges (or links) such as $e_{ih} = (v_i, v_h)$ $(v_i, v_h \in V)$. In the case that edge e_{ih} is allowed to traverse in only one direction (e.g., e_{ih} is allowed, but e_{hi} is not), G is called *directed graph*. When traverse in both directions is feasible, G is an *undirected graph*.

A document collection can be considered as a graph by regarding document d_i as vertex v_i. For instance, it is tentatively assumed that two documents are connected by an edge if they share 'two or more' different index terms. Under the assumption, the sample DB can be displayed as a graph in Figure 4.24, which is equivalently represented by a matrix

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{4.222}$$

This is an *adjacency matrix*, which is formally defined as $\mathbf{G} = [g_{ih}]$ such that

$$g_{ih} = \begin{cases} 1, & e_{ih} \in E \\ 0, & \text{otherwise} \end{cases}, \tag{4.223}$$

where $i, h = 1, \ldots, N$ (i.e., $\mathbf{G}$ is an $N \times N$ matrix) [83].

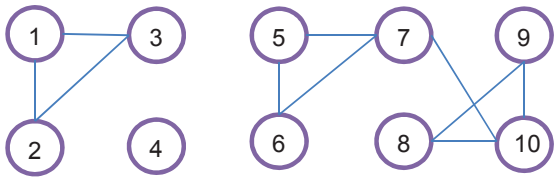


Figure 4.24: Graph representation of sample DB

Otherwise, it may be possible to use cosine measure between two documents as a weight of the edge, which constitutes a *weighted graph*. Actually, the similarity matrix of Table 3.3 is a lower triangle matrix of the weighted graph with removing diagonal elements, and therefor, a hierarchical clustering algorithm (Section 3.1.2) can be applied for detecting a structure inherent in the graph.

Min-cut clustering

Another strategy for identifying the graph structure is to partition directly a graph into subgraphs by removing edges [80]. Generally, if a subset E' of E includes only edges between two subsets V_1 and V_2 such that $V = \{V_1, V_2\}$, then E' is a *cut*, which separates the graph exclusively into two parts. For instance, in a graph shown as Figure 4.25, when edge e_{46} is removed, the graph is partitioned into $V_1 = \{v_1, v_2, v_3, v_4\}$ and $V_2 = \{v_5, v_6, v_7, v_8\}$ (note that $E' = \{e_{46}\}$, which is the cut).

[80] Formally, if $V' \subseteq V$ and $E' \subseteq E$, then $G' = \{V', E'\}$ is a *subgraph* of G (in other words, G is a *supergraph* of G') [83].

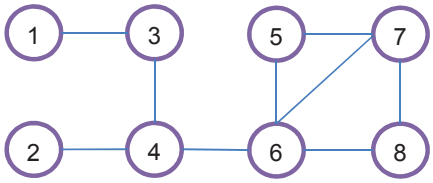


Figure 4.25: Example of graph

The size of a cut is measured as the sum of weights of edges on the cut, which is formally defined as

$$c(V_1, V_2) = \sum_{i:v_i \in V_1} \sum_{h:v_h \in V_2} s_{ih}, \tag{4.224}$$

where s_{ih} denotes the weight of edge e_{ih}. When $c(V_1, V_2)$ is calculated directly from an adjacency matrix $\mathbf{G}$ in Equation (4.222), it becomes that $s_{ih} = g_{ih}$. For partitioning a graph into two parts, it is natural to select first a cut whose size is minimal (i.e., 'min-cut'). This is often called *min-cut clustering* [214]. Actually, binary divisive clustering is feasible by partitioning the target document collection iteratively based on the min-cut [81].

Betweenness-based clustering

A *path* is a non-empty graph $P = \{V', E'\}$ such that

$$V' = \{v_1, v_2, \ldots, v_k\} \quad \textit{and} \quad E' = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)\},$$

where v_i $(i = 1, \ldots, k)$ are all distinct. Namely, vertices v_1 and v_k are linked by P [83]. The length of a path is measured as the number of edges on it. *Girvan-Newman algorithm* [107] is a graph-base clustering technique, which removes repeatedly an edge with the highest 'betweenness centrality' at the current stage. Actually, the betweenness centrality of an edge is measured by the number of shortest paths between nodes that go through the edge. A shortest path from v_i to v_h is denoted by $P_s(i, h)$ here. For example, in a graph shown in Figure 4.25, 16 shortest paths (i.e., $P_s(1, 5), P_s(1, 6), \ldots, P_s(4, 8)$) go through edge e_{46}, which leads to removing e_{46} first (as a result, the graph is partitioned into $\{v_1, v_2, v_3, v_4\}$ and $\{v_5, v_6, v_7, v_8\}$).

More precisely, steps of the Girvan-Newman algorithm are as follows:

1. Compute betweenness centralities of all edges.
2. Remove an edge with the largest betweenness centrality.
3. Terminate if there is no edge. Otherwise, recalculate the centralities and return to step 2.

By recording the removals of edges in each iteration, it is easy to construct a dendrogram after the procedure is terminated. For computing the betweenness centralities, it is necessary to identify the shortest paths between pairs of all vertices [82], which takes $O(Nm)$ times where $N = |V|$ and $m = |E|$ [214]. Because the procedure has to be repeated N times at most, the running time amounts to $O(N^2 m)$. Several modifications of the Girvan-Newman algorithm has been explored (see [173]).

4.3.2 Spectral clustering

Spectral graph partitioning

The min-cut method may choose uneven-sized partition (e.g., a single vertex may be separated from the rest of vertices). In order to avoid it and to obtain 'balanced' partition, 'ratio-cut' or

[81] As to an actual implementation of clustering algorithm based on the min-cut, see [92, 264], in which the method is called 'cut-clustering algorithm'.

[82] For instance, well-known *Dijkstra's algorithm* can be used for finding the shortest path (see [214]).

'normalized-cut' in which the size of each part is considered, are often used [214]. For instance, the ratio-cut is defined as

$$c_R(V_1, V_2) = \frac{c(V_1, V_2)}{|V_1|} + \frac{c(V_1, V_2)}{|V_2|}, \tag{4.225}$$

which includes the number of vertices in each part.

Spectral graph partitioning introduced in the 1970s [80] allows to find partition $V = \{V_1, V_2\}$ of an undirected graph so that its ratio-cut or normalized-cut is approximately minimum. The 'spectral' implies here 'spectrum' of a matrix, which is mathematically a set of its eigenvalues. In the spectral graph partitioning, the target is *Laplacian matrix* $\mathbf{L} = \mathbf{D} - \mathbf{S}$ where $\mathbf{D}$ is the diagonal 'degree' matrix with $[\mathbf{D}]_{ii} = \sum_h s_{ih}$, and $\mathbf{S} = [s_{ih}]$ (s_{ih} denotes the weight of edge e_{ih} as before). So, (i, h)-element of the Laplacian matrix is defined by

$$[\mathbf{L}]_{ih} = \begin{cases} \sum_{h'=1}^{N} s_{ih'}, & i = h \\ -s_{ih}, & i \neq h \end{cases} \tag{4.226}$$

(note that $s_{ih} = 0$ if $e_{ih} \notin E$ and that $s_{ii} = 0$). Based on the Laplacian matrix, the ratio-cut can be computed such that

$$\mathbf{q}^T \mathbf{L} \mathbf{q} = |V| \times c_R(V_1, V_2), \tag{4.227}$$

where $\mathbf{q}$ indicates an N-dimensional vector whose i-th element is defined by

$$q_i = \begin{cases} +\sqrt{|V_2|/|V_1|}, & v_i \in V_1 \\ -\sqrt{|V_1|/|V_2|}, & v_i \in V_2 \end{cases} \tag{4.228}$$

$(i = 1, \ldots, N)$ [83] [200].

For instance, when $N = 3$, since

$$\mathbf{q}^T \mathbf{L} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix} \begin{bmatrix} \sum_h s_{1h} & -s_{12} & -s_{13} \\ -s_{21} & \sum_h s_{2h} & -s_{23} \\ -s_{31} & -s_{32} & \sum_h s_{3h} \end{bmatrix}, \tag{4.229}$$

the value of $\mathbf{q}^T \mathbf{L} \mathbf{q}$ becomes

$$q_1 \left(q_1 \sum_h s_{1h} - q_2 s_{21} - q_3 s_{31} \right) + q_2 \left(-q_1 s_{12} + q_2 \sum_h s_{2h} - q_3 s_{32} \right) + q_3 \left(-q_1 s_{13} - q_2 s_{23} + q_3 \sum_h s_{3h} \right).$$

If it is supposed that $V_1 = \{v_1, v_2\}$ and $V_2 = \{v_3\}$, then

$$\mathbf{q}^T \mathbf{L} \mathbf{q} = (q_1^2 - q_1 q_3) s_{13} + (q_1^2 - q_1 q_3) s_{23} + (q_3^2 - q_1 q_3) s_{13} + (q_3^2 - q_1 q_3) s_{23}, \tag{4.230}$$

because $q_1 = q_2$ and $s_{ih} = s_{hi}$. Since $q_1^2 - q_1 q_3 = |V_2|/|V_1| + 1$ and $q_3^2 - q_1 q_3 = |V_1|/|V_2| + 1$,

$$\mathbf{q}^T \mathbf{L} \mathbf{q} = \frac{|V_1| + |V_2|}{|V_1|}(s_{13} + s_{23}) + \frac{|V_1| + |V_2|}{|V_2|}(s_{13} + s_{23}) = |V| \left(\frac{s_{13} + s_{23}}{|V_1|} + \frac{s_{13} + s_{23}}{|V_2|} \right). \tag{4.231}$$

Note that

$$\sum_{i=1}^{N} q_i = \sum_{i:v_i \in V_1} \sqrt{|V_2|/|V_1|} - \sum_{h:v_h \in V_2} \sqrt{|V_1|/|V_2|} = |V_1| \sqrt{|V_2|/|V_1|} - |V_2| \sqrt{|V_1|/|V_2|} = 0, \tag{4.232}$$

[83] Since $s_{ih} = s_{hi}$, it follows that

$$\mathbf{q}^T \mathbf{L} \mathbf{q} = \mathbf{q}^T \mathbf{D} \mathbf{q} - \mathbf{q}^T \mathbf{S} \mathbf{q} = \sum_{i=1}^{N} \sum_{h=1}^{N} s_{ih} q_i^2 - \sum_{i=1}^{N} \sum_{h=1}^{N} q_i q_h s_{ih} = \frac{1}{2} \sum_{i=1}^{N} \sum_{h=1}^{N} s_{ih} (q_i - q_h)^2.$$

By substituting Equation (4.228) into it,

$$\begin{aligned} \mathbf{q}^T \mathbf{L} \mathbf{q} &= \frac{1}{2} \sum_{i:v_i \in V_1} \sum_{h:v_h \in V_2} s_{ih} \left(\sqrt{|V_2|/|V_1|} + \sqrt{|V_1|/|V_2|} \right)^2 + \frac{1}{2} \sum_{i:v_i \in V_2} \sum_{h:v_h \in V_1} s_{ih} \left(-\sqrt{|V_1|/|V_2|} - \sqrt{|V_2|/|V_1|} \right)^2 \\ &= c(V_1, V_2) \left(|V_2|/|V_1| + |V_1|/|V_2| + 2 \right) = c(V_1, V_2) \left[(|V_1| + |V_2|)/|V_1| + (|V_1| + |V_2|)/|V_2| \right], \end{aligned}$$

can be obtained, and Equation (4.227) is derived from this result because $|V| = |V_1| + |V_2|$.

which is also represented $\mathbf{q}^T\mathbf{e} = 0$ where $\mathbf{e} = [1, 1, \ldots, 1]^T$, and that

$$\|\mathbf{q}\|^2 = \sum_{i=1}^{N} q_i^2 = |V_1|(|V_2|/|V_1|) + |V_2|(|V_1|/|V_2|) = |V_2| + |V_1| = N. \tag{4.233}$$

It is possible to compute vector $\mathbf{q}$ minimizing the criterion in Equation (4.227) under the constraints in Equations (4.232) and (4.233) based on an eigenvector corresponding to the second smallest eigenvalue of $\mathbf{L}$ (the minimum eigenvalue is zero) [200]. If $\mathbf{q}$ is obtained, then the vertices can be divided into two groups according to the sign of value of element q_i $(i = 1, \ldots, N)$. This is an approximation of the ratio-cut in the case of bipartition.

Partitioning into three or more subgraphs

In the case of partitioning V into L sets $\{V_1, V_2, \ldots, V_L\}$ where $L > 2$, the cut is written as

$$c(V_k, \bar{V}_k) = \sum_{i:v_i \in V_k} \sum_{h:v_h \in \bar{V}_k} s_{ih}, \tag{4.234}$$

where $\bar{V}_k = V \setminus V_k$ $(k = 1, \ldots, L)$. For partition $\{V_1, V_2, \ldots, V_L\}$, N-dimensional vector $\mathbf{b}_k$ is defined so that its i-th element becomes

$$b_{i|k} = \begin{cases} 1/\sqrt{|V_k|}, & \text{if } v_i \in V_k \\ 0, & \text{otherwise} \end{cases}, \quad k = 1, \ldots, L. \tag{4.235}$$

Since $\mathbf{b}_k^T\mathbf{L}\mathbf{b}_k = c(V_k, \bar{V}_k)/|V_k|$ for k-th set [84], the ratio-cut for L-partition can be computed as

$$c_R(V_1, \ldots, V_L) \equiv \sum_{k=1}^{L} \frac{c(V_k, \bar{V}_k)}{|V_k|} = \sum_{k=1}^{L} \mathbf{b}_k^T\mathbf{L}\mathbf{b}_k = \sum_{k=1}^{L} [\mathbf{B}^T\mathbf{L}\mathbf{B}]_{kk} = \mathrm{tr}(\mathbf{B}^T\mathbf{L}\mathbf{B}), \tag{4.236}$$

where $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_L]$ and $\mathbf{B}^T\mathbf{B} = \mathbf{I}$. So, minimization of $c_R(V_1, \ldots, V_L)$ equals to finding a partition $\{V_1, \ldots, V_L\}$ with the minimum of $\mathrm{tr}(\mathbf{B}^T\mathbf{L}\mathbf{B})$, and the minimization problem can be approximately solved by replacing $\mathbf{B}$ by eigenvectors of the Laplacian matrix $\mathbf{L}$ [85]. Consequently, a spectral clustering algorithm based on the criterion of minimizing the ratio-cut can be executed as shown in Figure 4.26. At last stage in the procedure, a k-means algorithm is applied for determining subgraphs (i.e., clusters) to which each vertex (i.e., document) belongs because discrete partition in Equation (4.235) has to be estimated from real numbers of the eigenvectors (note that each eigenvector $\mathbf{u}_k$ is used for $\mathbf{b}_k$, respectively).

In the sample DB, eigenvalues of the Laplacian matrix can be computed as

$$0.000, 0.696, 1.725, 2.024, 2.117, 3.342, 3.465, 3.861, 4.227, 4.744$$

in ascending order when the similarity matrix is constructed from tf vectors based on the cosine coefficient. If $L = 3$, then $L' = \lceil \log_2 3 \rceil = 2$ [86] and

$$\mathbf{u}_2 = [0.640, 0.354, 0.334, -0.213, 0.026, -0.152, -0.092, -0.270, -0.257, -0.370]^T,$$

$$\mathbf{u}_3 = [0.595, -0.256, -0.325, -0.164, -0.229, -0.069, -0.246, 0.084, 0.044, 0.567]^T,$$

[84] As before, from manipulation on $\mathbf{L}$,

$$\mathbf{b}_k^T\mathbf{L}\mathbf{b}_k = \frac{1}{2}\sum_{i=1}^{N}\sum_{h=1}^{N} s_{ih}(b_{i|k} - b_{h|k})^2,$$

can be obtained. If $v_h \notin V_k$, then $b_{h|k} = 0$ and only $s_{ih}/|V_k|$ remains in the sum. Otherwise, $(b_{i|k} - b_{h|k})$ becomes zero.

[85] Let $\mathbf{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_L]$ be a diagonal matrix of the Lagrange multipliers. The object function is $\mathcal{H} = \mathrm{tr}(\mathbf{B}^T\mathbf{L}\mathbf{B}) - \mathbf{\Lambda}[\mathbf{B}^T\mathbf{B} - \mathbf{I}]$, and $\partial\mathcal{H}/\partial\mathbf{B} = \mathbf{L}\mathbf{B} - \mathbf{\Lambda}\mathbf{B}$. Thus $\partial\mathcal{H}/\partial\mathbf{B} = \mathbf{0}$ means that $\mathbf{B}$ is a matrix of eigenvectors for eigenvalues in $\mathbf{\Lambda}$.

[86] $\lceil x \rceil$ is called ceiling function, which indicates a minimum integer that be greater than x. Note that $\log_2 3$ is about 1.58.

‡ Spectral clustering based on minimizing the ratio-cut

Set: The number of clusters L and similarity matrix $\mathbf{S} = [s_{ih}]$

1) Compute the Laplacian matrix $\mathbf{L}$.
2) Compute the first $L'+1$ eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_{L'+1}$ of $\mathbf{L}$ (e.g., $L' = \lceil \log_2 L \rceil$ [80]).
3) Extract i-th row of matrix $\tilde{\mathbf{U}} = [\mathbf{u}_2, \ldots, \mathbf{u}_{L'+1}]$ as new representation $\mathbf{x}_i^T$ ($i = 1, \ldots, N$).
4) Cluster $\mathbf{x}_1, \ldots, \mathbf{x}_N$ by a k-means algorithm.

Out: Clusters $C_1, \ldots, C_L$.

Figure 4.26: Spectral clustering algorithm based on the criterion of minimizing the ratio-cut

which leads to a cluster set of $C_1 = \{d_1\}$, $C_2 = \{d_2, d_3, d_5, d_7\}$ and $C_3 = \{d_4, d_6, d_8, d_9, d_{10}\}$ by using the the Hartigan-Wong algorithm.

Meanwhile, the normalized-cut is represented as

$$c_N(V_1, \ldots, V_L) = \sum_{k=1}^{L} \frac{c(V_k, \bar{V}_k)}{d(V_k)}, \tag{4.237}$$

where $d(V_k) \equiv \sum_{i:v_i \in V_k} [\mathbf{D}]_{ii}$ (i.e., the sum of edge weights). For approximating the criterion, the Laplacian matrix has to be normalized, for instance,

$$\mathbf{L}_n = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2} \tag{4.238}$$

(see [200] for details) [87]. When using the criterion, the partition is often referred to as *normalized spectral clustering*. In contrast, clustering based on the raito-cut using straightforwardly the Laplacian matrix in Figure 4.26 can be called *unnormalized spectral clustering*. In the case of DC, like hierarchical clustering, it is often hard to compute the similarity matrix including $O(N^2)$ pairs of documents when executing the spectral clustering. In addition, eigenvalues and eigenvectors have to be computed for a large-scale Laplacian matrix if N is large.

Spectral co-clustering

Dhillon(2001) [80] applied the spectral clustering algorithm to an undirected 'bipartite' graph including both documents and terms as its vertices. Generally, a graph is called 'r-partite' when V is partitioned into r classes such that every edge has its ends in different classes and that vertices in the same partition class must not be adjacent [83]. In Dhillon(2001) [80], one class is the document set and the other is the word set, and its weighted adjacency matrix is constructed as

$$\mathbf{M} = \begin{bmatrix} \mathbf{O} & \mathbf{W}^T \\ \mathbf{W} & \mathbf{O} \end{bmatrix}, \tag{4.239}$$

where $\mathbf{W}$ is a document-by-term matrix as before. So, $\mathbf{M}$ is an $(M+N) \times (M+N)$ matrix and its partitioning allows for grouping documents and terms simultaneously, which is often called *co-clustering*.

Actually, Dhillon(2001) [80] employed a more convenient formula based on SVD,

$$\mathbf{W}_n = \mathbf{U}\mathbf{Q}\mathbf{V}^T, \tag{4.240}$$

where $\mathbf{W}_n = \mathbf{D}_2^{-1/2}\mathbf{W}\mathbf{D}_1^{-1/2}$ and

$$\mathbf{D} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_2 \end{bmatrix}, \tag{4.241}$$

[87] Note that i-th element of $\mathbf{b}_k$ is 1 if $v_i \in V_k$, otherwise zero, in this instance

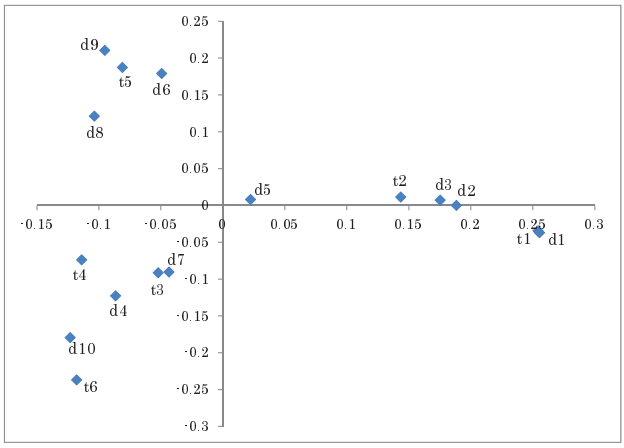


Figure 4.27: Result of spectral co-clustering for the sample DB

which can be derived by some algebraic manipulations under an assumption that normalized-cut is minimized [88]. In this case, the data entering into the k-means algorithm are

$$\mathbf{X} = \begin{bmatrix} \mathbf{D}_1^{-1/2}\tilde{\mathbf{V}} \\ \mathbf{D}_2^{-1/2}\tilde{\mathbf{U}} \end{bmatrix}, \tag{4.242}$$

where $\tilde{\mathbf{V}} = [\mathbf{v}_2, \ldots, \mathbf{v}_{L'+1}]$ and $\tilde{\mathbf{U}} = [\mathbf{u}_2, \ldots, \mathbf{u}_{L'+1}]$. Note that vectors $\mathbf{v}_h$ and $\mathbf{u}_h$ are arranged in ascending order of $(1-\eta_h)$ where η_h is h-th diagonal element of matrix $\mathbf{Q}$ ($h = 1, \ldots, \min(N, M)$). Actually, for the sample DB,

$$\mathbf{W}_n = \begin{bmatrix} 1.789 & 0.447 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.500 & 1.500 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.378 & 2.268 & 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.000 & 1.512 & 1.134 & 0.000 & 0.000 \\ 0.000 & 0.894 & 0.894 & 0.000 & 0.447 & 0.000 \\ 0.000 & 0.378 & 0.378 & 0.000 & 1.890 & 0.000 \\ 0.000 & 0.354 & 1.768 & 0.000 & 0.354 & 0.354 \\ 0.000 & 0.000 & 0.000 & 0.894 & 1.342 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.408 & 2.041 & 0.000 \\ 0.000 & 0.000 & 0.000 & 1.061 & 0.354 & 1.414 \end{bmatrix}, \tag{4.243}$$

and when $L = 3$, a cluster set,

$$C_1 = \{d_1, d_2, d_3, d_5, t_1, t_2\}, C_2 = \{d_4, d_7, d_{10}, t_3, t_4, t_6\}, C_3 = \{d_6, d_8, d_9, t_5\},$$

was obtained by the Hartigan-Wong algorithm from $\mathbf{X}$ which is an $(M + N) \times 2$ matrix (i.e., $L' = 2$). Figure 4.27 is a plot of documents and terms where the first column of $\mathbf{X}$ is used on the x-axis and the second column on the y-axis.

[88] From Equation (4.238), the eigenvalue problem for minimizing the normalized-cut is represented by $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}\mathbf{z} = \lambda\mathbf{z}$ where $\mathbf{z}$ indicates an eigenvector and λ is an eigenvalue. In the case of the bipartite graph represented in Equation (4.239), since $\mathbf{L} = \mathbf{D} - \mathbf{M}$, it becomes that

$$\begin{bmatrix} \mathbf{D}_1 & -\mathbf{W}^T \\ -\mathbf{W} & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \lambda \begin{bmatrix} \mathbf{D}_1 & \mathbf{O} \\ \mathbf{O} & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix},$$

where $\mathbf{z} = [\mathbf{x}^T, \mathbf{y}^T]^T$. The equation is separated into $\mathbf{D}_1\mathbf{x} - \mathbf{W}^T\mathbf{y} = \lambda\mathbf{D}_1\mathbf{x}$ and $-\mathbf{W}\mathbf{x} + \mathbf{D}_2\mathbf{y} = \lambda\mathbf{D}_2\mathbf{y}$, which can be rewritten such as $\mathbf{v} - \mathbf{D}_1^{-1/2}\mathbf{W}^T\mathbf{D}_2^{-1/2}\mathbf{u} = \lambda\mathbf{v}$ and $-\mathbf{D}_2^{-1/2}\mathbf{W}\mathbf{D}_1^{-1/2}\mathbf{v} + \mathbf{u} = \lambda\mathbf{u}$ if putting that $\mathbf{v} = \mathbf{D}_1^{1/2}\mathbf{x}$ and $\mathbf{u} = \mathbf{D}_2^{1/2}\mathbf{y}$. These equations imply that $\mathbf{W}_n^T = (1-\lambda)\mathbf{v}\mathbf{u}^T$ and $\mathbf{W}_n = (1-\lambda)\mathbf{u}\mathbf{v}^T$, which are equivalent to SVD in Equation (4.240). Thus after computing the SVD, the eigenvector can be obtained as $[[\mathbf{D}_1^{-1/2}\mathbf{v}]^T, [\mathbf{D}_2^{-1/2}\mathbf{u}]^T]^T$.

Chapter 5

Experiments of Large-Scale Multilingual Document Clustering

5.1 Scalable Multilingual Document Clustering Technique

Through discussions in Chapters 2, 3 and 4, characteristics or properties of techniques of cross-language information retrieval (CLIR) and document clustering (DC) have become sufficiently clear. Based on them, this section attempts to propose a method for large-scale multilingual document clustering (MLDC).

5.1.1 Literature on Multilingual Document Clustering

When the target multilingual set of documents is sufficiently small and the length of each document is short, machine translation (MT) software can be used for translating the documents into a single language. In this case, MLDC reduces to standard monolingual DC. For example, Rauber et al.(2001) [248] applied an algorithm of self-organizing map (SOM) (Section 3.3.5) to a set of documents translated by MT software. MT software was also used in Columbia Newsblaster [183, 89], which is a system for summarizing news articles collected from the web.

In Topic Detection and Tracking (TDT) evaluation workshop [14] (Section 3.3.7), many research groups have used results of MT, which were provided as a part of TDT corpus, for clustering English and Chinese news stories. The collection of stories used in this workshop was not small, which implies that MT is applicable to large multilingual sets. A typical method of MLDC in TDT workshop would be incremental one (e.g., [182]), in which each translated story is allocated to a cluster whose centroid (see Equation (3.6)) is the most similar.

If MT software is not available, then bilingual dictionaries, multilingual thesauri or parallel corpora can be employed as language resources for translation (Section 2.2.3). In Pouliquen et al.(2004) [241], documents in different languages were mapped onto a single classification scheme by using a multilingual thesaurus Eurovoc, and similarly, Montavalo et al.(2007) [216] used Euro Wordnet. Also, Pham et al.(2008) [235] have applied a language-independent ontology for SOM-based clustering of multilingual medical documents. In contrast, parallel corpora were used as language resources in Chau et al.(2005) [56] and Wei et al.(2008) [314]. Chau et al.(2005) [56] developed a technique of supervised text categorization enabling unsupervised document clustering to be executed in a multilingual setting, in which language-independent concept-based vectors representing multilingual documents were generated by applying an SOM algorithm to a parallel corpus, and as a result, a hierarchy of document categories was constructed based on the vectors. In Wei et al.(2008) [314], a latent semantic indexing (LSI) technique was applied to MLDC. More specifically, agglomerative hierarchical clustering was executed based on document vectors in a multilingual semantic space constructed from a parallel corpus using singular value decomposition (SVD) (Section 2.2.2).

Silva et al.(2001) [283] proposed a method of dividing a document collection based on model-based clustering analysis with multivariate Gaussian mixtures (Section 4.1.1), in which the doc-

ument set is assumed to be a parallel corpus. Dictionary-based translation (Section 2.2.3) was adopted by Mathieu et al.(2004) [207], Wu & Lu.(2007) [319] and so on. Mathieu et al.(2004) [207] applied shared nearest neighbor (SNN) to a multilingual set of documents in which similarities were measured based on dictionary-based translation with no disambiguation. Wu & Lu(2007) [319] proposed a unique method of determining concurrently clusters and optimal translations describing the content of clusters based on a domain alignment translation model. When there are source terms having no corresponding translation in language resources, cognate matching (Section 2.2.2) can be used to measure the similarities between documents in different languages. Montavalo et al.(2007) [216] attempted to compare MLDC performance between term translation and cognate matching of named entities based on edit distance (for named entities, see also [241]). Meanwhile, in Chen & Lin(2000) [60] and Chen & Ku(2002) [61], a transliteration method (Section 2.2.2) was used for clustering a set of English and Chinese documents.

5.1.2 Leader-follower algorithm with dictionary-based translation

For clustering large-scale multilingual document sets, it is indispensable to adopt the following two strategies as components of the system, namely,

1. dictionary-based translation, and
2. leader-follower clustering (LFC) algorithm,

which is a straightforward conclusion from discussions in Chapters 2, 3 and 4. First, in the case of MLDC, document translation (Section 2.2.2) has to be attempted, and the dictionary-based translation would be optimal when the document set is large. In this thesis, techniques based on parallel corpora are out of the scope because their applicability is very limited by availability of such special corpora. In practice, various languages may be processed in MLDC situations, and it is unrealistic to assume the availability of parallel corpora for them. Second, the LFC algorithm can be considered as a more promising clustering technique for large document sets than any other methods. It is hard to apply probabilistic mixture models (Section 4.1.1) to situations in which the total number of distinct terms M is so large. For instance, in multinomial mixture model (MMM), the probability is computed by Equation (4.28) including multiplication $\prod_{j=1}^{M}$, and the calculation by computers would easily underflow to zero when M is large. Similarly, matrix-based clustering methods (Section 4.2) would not be appropriate for large-scale document clustering due to the fact that a very large term-by-document matrix has to be processed many times. For example, nonnegative matrix factorization (NMF) (Section 4.2.4) requires many matrix computations for estimating the factorization (see Equations (4.198) and (4.199)).

However, the simple and 'powerful' combination of the dictionary-based translation and the LFC algorithm for MLDC has not been sufficiently verified in literature so far [1]. Therefore, it is indispensable to verify experimentally 'effectiveness' of the 'efficient' method in the following points:

1. Effectiveness of scalable technique for translation disambiguation used in the dictionary-based translation.
2. Effectiveness of the LFC algorithm for monolingual DC in comparison with more theoretically sophisticated techniques.
3. Effectiveness of the LFC algorithm with dictionary-based translation for large-scale MLDC.

The following sections in this chapter will describe three experiments for examining the three kinds of effectiveness, respectively.

[1] Kishida(2011) [154] reported exceptionally an experiment exploring the combination of two methods, which is a main part of this thesis and will be discussed in Section 5.4.

5.2 Experiment of Term Disambiguation Techniques

This section reports an experiment on term disambiguation techniques for CLIR based on Kishida (2007) [150]. As discussed in Section 2.2.3, dictionary-based translation would yield many irrelevant terms in the context of individual documents, and therefore, translation disambiguation may play an important role for MLDC.

5.2.1 Purpose and methods

Since translation disambiguation techniques based on the target document sets do not require special language resources (e.g., parallel corpus), they are more appropriate for the purpose of this thesis than the other techniques discussed in Section 2.2.4 from a practical viewpoint. In order to examine empirically performance of the techniques, a standard experiment of bilingual retrieval from German to Italian was attempted by using the CLEF 2003 test collection [2] (note that this experiment is an IR test, not DC test, unlike those reported in the following sections). The target Italian document collection in this experiment is a set of full-text news articles (see [39]).

The method adopted for executing CLIR is basically dictionary-based query translation, and English was used as an intermediary language for translating German into Italian. This is often called 'pivot language approach' (Section 2.2.3), namely, the translation proceeds as 'German terms → English translations → Italian translations'. More precisely, first, each query term in German is replaced by a set of the corresponding English terms based on a German to English dictionary, and second, each term in the English set is again replaced by a set of the corresponding Italian terms according to an English to Italian dictionary. Such transitive translation via a pivot language often yields many irrelevant translations because all final translations obtained from irrelevant terms in the intermediary language (i.e., English) are possibly also irrelevant (see Figure 2.5). Therefore, it is especially important for the dictionary-based pivot language approach to enhance search performance by incorporating translation disambiguation.

Furthermore, in order to verify research findings from the German to Italian searches, the similar experiment was repeated for the case of French to Italian (pivot language approach via English) and English to Italian (non-pivot) searches on the same CLEF 2003 test collection. The additional experiments will be described after discussing results from the experiment of German to Italian searches.

The experiment compared search performance between the following term disambiguation techniques:

1. Disambiguation by term co-occurrence statistics (see Figure 2.7)
   (a) Best pairs selection based on Equation (2.55)
   (b) Best sequence selection based on Equation (2.57)
   (c) Best cohesion selection based on Equation (2.60)
2. PRF based disambiguation by Kishida & Kando(2004) [157] according to Equation (2.62)

Note that they are techniques using only the target document collection.

At the time that an index file for the original Italian document collection was created, term co-occurrence was also counted and recorded in another index file enabling to find quickly information on the occurrence. In this experiment, sentence-based co-occurrence was adopted, namely, sentences in which a pair of terms appear together were counted and used as 'the number of co-occurrences'. Similarly in this experiment, 'the number of occurrences' indicates the number of sentences including a given term, which denoted by $c(t)$ where t is an index term. As similarity measures of two terms computed from co-occurrence frequencies, the experiment used the mutual information (MI), cosine coefficient, Dice coefficient, and overlap coefficient, which are defined as

$$\log_2 \frac{Sc(t,s)}{c(t)c(s)}, \quad \frac{c(t,s)}{\sqrt{c(t)c(s)}}, \quad \frac{2c(t,s)}{c(t)+c(s)}, \quad \frac{c(t,s)}{\min[c(t),c(s)]},$$

[2] CLEF (Cross-Language Evaluation Forum) is a well-known research project for CLIR experiments. At present, it is named 'Conference and Labs of the Evaluation Forum' (see http://www.clef-initiative.eu//).

respectively, where $c(t, s)$ indicates the number of sentences including both terms t and s, and S is the total number of sentences in the target collection (see Section 2.2.4).

5.2.2 Search algorithm and text processing

The well-known BM25 of Okapi formula in Equation (2.28) [259] was used for computing document scores in this experiment. In order to compute the scores for a given query, both German and Italian text in queries and documents were processed by the following steps: 1) identifying tokens, 2) removing stopwords, 3) stemming (Section 2.1.6). In particular, decomposition of compound words in German was executed using a heuristic rule based on longest matching against headwords included in a German to English dictionary, according to Kishida & Kando(2004) [157]. For example, German word, "Briefbombe" is broken down into two headwords listed in the German to English dictionary, "Brief" and "Bombe", according to a rule that only the longest headwords included in the original compound word are extracted from it (see also Section 5.4.2 for decomposition of German compound words).

Free bilingual dictionaries (German to English and English to Italian) on the Internet [3] were used for transitive query translation. Stemmers and stopword lists for German and Italian languages were downloaded from the web site of the Snowball project [4]. Stemming for English was performed by the original Porter's algorithm [240]. Before executing transitive translation by using two bilingual dictionaries, all terms included in the bilingual dictionaries were normalized according to the same stemming procedure for processing text of documents and queries. The actual translation process was a simple replacement. If no corresponding headword was included in the dictionaries (German-English or English-Italian), then the unknown word was sent directly to next step without any change.

5.2.3 Results and analysis

Effectiveness of similarity measures

The Italian document collection contains 157,558 documents in total, and its average document length became 181.86 tokens after text processing. First of all, the effectiveness of similarity measures (i.e., MI, cosine, Dice and overlap) used in the process of the term co-occurrence based method for disambiguation is discussed here. For comparison of the effectiveness of searches, scores of mean average precision (MAP) (Section 2.1.8) were calculated for a set of 51 search topics having one or more relevant documents in the Italian document collection (see Table 5.1). In Table 5.1, MAP scores are listed by 'short queries' and 'long queries'. In the case of short queries, each search was executed using only the <TITLE>field in the topics of the CLEF 2003 test collection, whereas in the case of long queries, both the <TITLE>and <DESCRIPTION>fields were employed. The <TITLE>field usually includes a few search terms, and a sentence representing the search query is described in the <DESCRIPTION>field (see [39]). Since the sentence was decomposed according to the procedure described above, each long query tends to contain relatively many terms as a result. Therefore, unfortunately, search results for the 'best sequence' algorithm in the case of long queries could not be obtained in this experiment due to excessive computational complexity.

The experiment also attempted post-translation feedback [15] (Section 2.2.4), which is often used for enhancing search performance after the process of translation disambiguation. In this experiment, a standard PRF technique (Section 2.1.4) was simply used. Actually, the feedback was executed by extracting 10 terms from top-ranked 30 documents, which were obtained by an initial search using translations selected by each disambiguation method, based on term weight ω_j in Equation (2.36) [5]. However, from the purpose of this thesis, the effectiveness of disambiguation methods without the post-translation feedback should be mainly discussed because other factors would affect the search performance through the feedback. Therefore, the search results with the post-translation feedback will be treated as supplementary findings in this thesis.

[3] http://www.freelang.net/

[4] http://snowball.tartarus.org/

[5] According to a standard manner, if an extracted term was already included in the set of initial query terms, then its weight in the query was increased to 1.5 times. Otherwise, its weight was set to be 0.5 (see Section 2.1.5).

Table 5.1: MAP scores for term co-occurrence based methods

	MI		Dice		Cosine		Overlap	
	Short queries	Long queries	Short queries	Long queries	Short queries	Long queries	Short queries	Long queries
Without feedback								
Best pair	0.1430	0.1516	0.1619	0.1941	0.1558	0.1967	0.1539	0.1786
Best cohesion	0.1514	0.1857	0.1620	0.1988	0.1601	0.2031	0.1565	0.1881
Best sequence	0.1747	N/A	0.1621	N/A	0.1796	N/A	0.1550	N/A
Average	0.1564	0.1687	0.1620	0.1964	0.1652	0.1999	0.1551	0.1834
	0.1613		0.1758		0.1790		0.1664	
With feedback								
Best pair	0.1462	0.1967	0.1687	0.2308	0.1545	0.2223	0.1587	0.1984
Best cohesion	0.1480	0.2239	0.1685	0.2281	0.1636	0.2398	0.1587	0.2075
Best sequence	0.1812	N/A	0.1700	N/A	0.1893	N/A	0.1599	N/A
Average	0.1585	0.2130	0.1691	0.2294	0.1691	0.2310	0.1591	0.2029
	0.1792		0.1932		0.1939		0.1766	

N/A: Results could not be obtained due to excessive computational complexity.

In Table 5.1, it can be observed that the cosine coefficient provided better search results than any other similarity measures, namely, the best performance was obtained by using the cosine coefficient in all cases of searching for short queries without feedback (its MAP score is 0.1796), short queries with feedback (0.1893), long queries without feedback (0.2031) and long queries with feedback (0.2398). The averages of MAP scores for the cosine coefficient were also higher than those for other measures (0.1790 without feedback and 0.1939 with feedback). However, it should be noted that the Dice coefficient was consistently more effective than the cosine measure in the cases of the best pair and best cohesion methods for short queries, namely, the cosine coefficient was not always dominant in every case.

It should be also noted that the difference of scores was not so large for making a definite conclusion on the dominance of the cosine coefficient. For example, the difference of MAP scores between the cosine and MI was only 0.0049 in the case of best sequence algorithm for short queries without feedback (i.e., 0.1796 for the cosine and 0.1747 for MI). Therefore, a statistical test is needed for confirming reliability of the conclusions, and actually, the sign test, which is a non-parametric method for statistically checking if two samples were drawn from a single population (Section 2.1.8), was used. In this experiment, $H = 51$ (i.e., the total number of topics), and the total number of topics excluding those where two average precision values are equal (i.e., 'tie') is denoted by H' (note that m means the number of topics where one method outperforms the other, as before).

Table 5.2 shows the probabilities between the best two results for two cases of short queries and long queries without feedback. In both the cases, values of H' are small (i.e., 14 and 10, respectively, whereas the total number of topics is 51). This means that an identical set of translations was selected in most of the topics. For the case of short queries, the cosine coefficient outperforms the MI in 11 out of 14 topics, and the probability is 0.0287, which is a statistically significant difference at 5% level. Although the difference of MAP scores is very small (i.e., 0.0049), it can be concluded that there is statistically significant difference due to the fact that the cosine coefficient was dominant in many topics. In contrast, for the case of long queries, the cosine coefficient was dominant in only 4 out of 10 topics, which means no statistically significant difference. Hence, it is not feasible to draw a clear conclusion on the dominance of the cosine coefficient for long queries containing many terms.

Selection algorithms for term co-occurrence based method

As discussed in Section 2.2.4, the performance of selecting translations based on the best pairs tends to be relatively poor in Table 5.1. There is no case in which the MAP score of this algorithm is greater than that of the best cohesion or the best sequence methods (except the results with

Table 5.2: Results of sign tests (1): similarity measures

(a) short queries without feedback			(b) long queries without feedback		
Cosine	/sequence	0.1796	Cosine	/cohesion	0.2031
MI	/sequence	0.1747	Dice	/cohesion	0.1988
Difference		0.0049	Difference		0.0043
Probability		0.0287	Probability		0.8281
		$H' = 14, m = 11$			$H' = 10, m = 4$

Table 5.3: Average of MAP scores by disambiguation method

Algorithms	Short queries	Long queries
Best pair	0.1536	0.1803
Best cohesion	0.1575	0.1939
Best sequence	0.2132	N/A

Note: without feedback

feedback) although the difference is small. Also, the best sequence algorithm is dominant except just one case of the overlap coefficient for short queries (the best cohesion outperforms the best sequence in this case).

From these empirical observations, the relative performance can be represented as

$$\text{Best pair} \prec \text{Best cohesion} \prec \text{Best sequence},$$

which is almost consistent with the conclusion derived logically in Section 2.2.4. This is again observed in Table 5.3, which shows averages of the MAP scores for each disambiguation technique. The highest average for short queries was that of the best sequence algorithm, followed in order by the best cohesion and the best pair. For long queries, the average for the best pair algorithm was lower than that for the best cohesion.

Results of the sign test is shown in Table 5.4 for only cases using the Dice or cosine measure without feedback. Although there is no statistically significant difference between the term selection algorithms (i.e., all probabilities are over 0.05), it turns out that the best sequence algorithm outperforms the best pair method, namely, probabilities of the best sequence against the best pairs are 0.3438 (Dice) and 0.1133 (cosine), which are relatively small. For example, the probability 0.1133 was obtained from a binomial distribution with $m = 8$ and $H' = 11$. This means that there are 8 topics for which the one method outperforms, 3 topics for which the other outperform, and 40 topics in which the performance is equal (i.e., 'ties'). A statistically significant difference may be observed if a larger sample is available. In contrast, the evidence that the best sequence outperforms the best cohesion may not be enough, since the probabilities for Dice and cosine are 0.6563 (Dice) and 0.3770 (cosine), respectively. With regard to relationship between the best cohesion and the best pair algorithms, only in the cases of long queries with Dice and short queries with cosine, it may be concluded that the best cohesion algorithm yields better results than the best pair method, although no statistically significant difference was observed.

Comparison of term co-occurrence and PRF based methods

MAP scores of search runs using PRF based methods are shown in Table 5.5, in which the number of top-ranked documents that are assumed to be relevant is set to be 30, 50, 100, 500 and 1000, respectively (the numbers of top-ranked documents were arbitrarily selected in order to look for the best-possible performance). It can be seen that the performance is slightly changed by the number of top-ranked documents used for disambiguation. On average, it seems that the case of top-ranked 30 documents was dominant in the situation without feedback, but its degree of dominance was small.

Table 5.6 summarizes the best scores selected from Tables 5.1 and 5.5, respectively. As indicated, the term co-occurrence based method (with the best sequence algorithm) outperforms

Table 5.4: Result of sign tests (2): term co-occurrence based methods

Similarity measure	Queries	Method 1 & MAP	Method 2 & MAP	H'	m	$P(x \geq m)$
Dice	Short	Sequence: 0.1621	Pair: 0.1619	6	4	0.3438
	Short	Sequence: 0.1621	Cohesion: 0.1620	6	3	0.6563
	Short	Cohesion: 0.1620	Pair: 0.1619	2	2	0.2500
	Long	Cohesion: 0.1988	Pair: 0.1941	19	12	0.1796
Cosine	Short	Sequence: 0.1796	Pair: 0.1558	11	8	0.1133
	Short	Sequence: 0.1796	Cohesion: 0.1601	10	6	0.3770
	Short	Cohesion: 0.1601	Pair: 0.1558	5	4	0.1875
	Long	Cohesion: 0.2031	Pair: 0.1967	20	9	0.7483

Note: without feedback

Table 5.5: MAP scores for PRF based methods

	No. of top-ranked documents				
	30	50	100	500	1000
Without feedback					
Short queries	0.1690	0.1552	0.1508	0.1489	0.1639
Long queries	0.2073	0.2037	0.2096	0.2059	0.2044
Average	0.1882	0.1794	0.1802	0.1774	0.1841
With feedback					
Short queries	0.1885	0.1737	0.1724	0.1670	0.1826
Long queries	0.2496	0.2605	0.2686	0.2642	0.2507
Average	0.2191	0.2171	0.2205	0.2156	0.2166

the PRF method for short queries whereas the PRF method outperforms the term co-occurrence based method for long queries. This empirical observation suggests that the best sequence algorithm provides higher performance, but unfortunately it is not feasible to use it due to excessive computational complexity in some situations as discussed above (actually, the algorithm was not able to work for long queries in this experiment). If the computation is impossible, then there is no alternative than using the best cohesion algorithm. It is therefore worth comparing the performance between the best cohesion and the PRF based method. As shown in Table 5.6, the PRF based method outperforms the best cohesion algorithm for both short and long queries. However, no statistically significant difference was observed between these methods (see Table 5.7).

Table 5.6 also shows MAP scores of search results without any disambiguation, which can be considered as a 'baseline'. Clearly, there are large differences of the scores between the baseline and the disambiguation methods. Actually, the sign tests indicate statistically significant differences at 1% level between the results with disambiguation and with no disambiguation. For instance, its probability is 0.0008 between the best sequence with the cosine and no disambiguation for short queries (without feedback) and is 0.0010 between the PRF based method by the top-ranked 100 documents and no disambiguation for long queries (without feedback). Finally, recall-precision curves of search runs listed in Table 5.6 are shown in Figures 5.1 and 5.2 for short and long queries, respectively.

Additional experiments for verifying research results

In order to confirm reliability of results obtained from the empirical test on German to Italian searches, the same experiment was repeated for French to Italian and English to Italian searches by using the same test collection (CLEF 2003). French topics were transitively translated into Italian by the pivot approach via English like German to Italian searches, but inevitably, English to Italian bilingual retrieval was non-pivot. Procedure of searches was almost similar to that in the German to Italian case, and language resources (French to English dictionary, French stopword list and French stemmer) were also downloaded from the same web sites used for German to Italian

Table 5.6: Comparison of MAP scores (summary)

Methods	Without feedback	With feedback
1) short queries: TITLE only		
No disambiguation	0.1344	0.1537
Best pair	0.1619 (Dice)	0.1687 (Dice)
Best cohesion	0.1620 (Dice)	0.1685 (Dice)
Best sequence	0.1796 (Cosine)	0.1893 (Cosine)
PRF	0.1690 (top 30)	0.1885 (top 30)
(2) long queries: TITLE and DESCRIPTION		
No disambiguation	0.1484	0.1537
Best pair	0.1967 (Cosine)	0.2308 (Dice)
Best cohesion	0.2031 (Cosine)	0.2398 (Cosine)
Best sequence	N/A	N/A
PRF	0.2096 (top 100)	0.2686 (top 100)

Table 5.7: Result of sign tests (3): term co-occurrence and PRF based methods

	Method 1	Method 2	H'	m	Prob.
Short queries	PRF (top 30)	Dice (Cohesion)	23	11	0.6612
Long queries	PRF (top 100)	Cosine (Cohesion)	35	17	0.6321

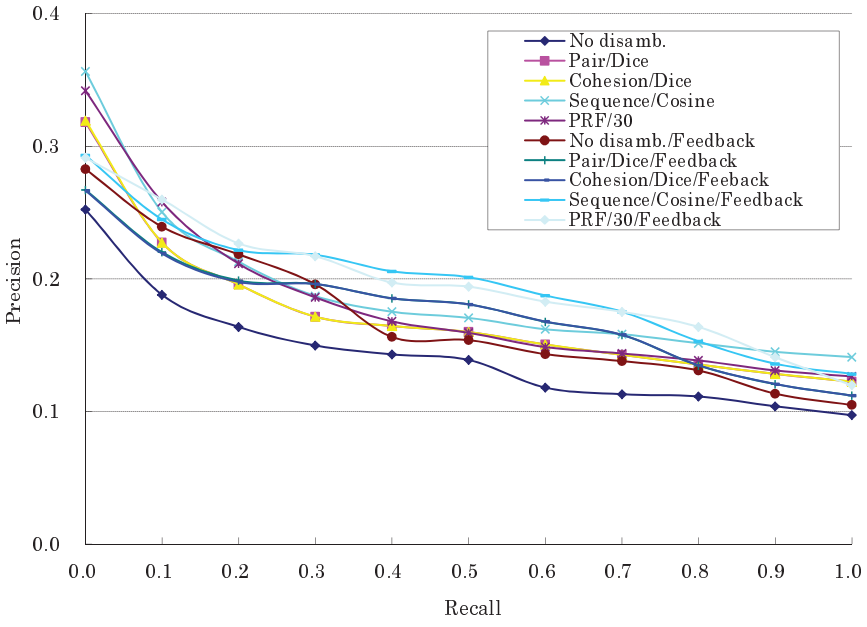


Figure 5.1: Recall-precision curves for short queries

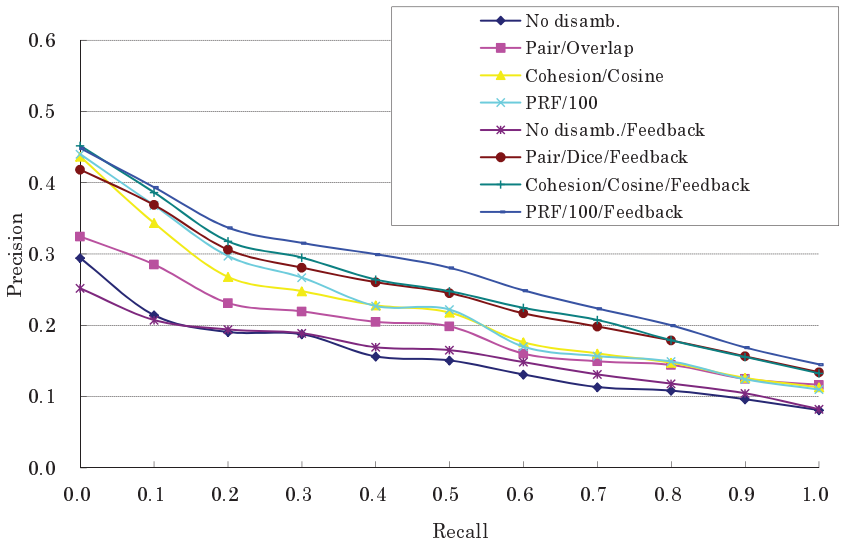


Figure 5.2: Recall-precision curves for long queries

retrieval. However, unlike text processing for German topics, any term in French and English topics was not decomposed.

Results of additional experiments were almost similar to that in the case of German to Italian retrieval. MAP scores of searches for short queries with no feedback are shown in Table 5.8. First, it can be concluded again that the dominant method for disambiguation based on term co-occurrence was to select the best sequence since its average scores were the highest in both cases of French and English topics. The best sequence method is followed by the best cohesion (see Table 5.8). Second, the PRF method showed again good performance. Especially, in these additional experiments, this method outperforms slightly the term co-occurrence based methods ('Top 100' and 'Top 50' were dominant among all runs of French to Italian and English to Italian retrieval, respectively).

Finally, it was observed that the cosine similarity is superior to other measures in the case of French to Italian retrieval. Also, the Dice coefficient outperforms the cosine coefficient in the non-pivot case (English to Italian) and the dominance of the Dice coefficient was also partly observed in the case of German to Italian searches although the differences were so small (see Table 5.1) [6].

5.2.4 Discussion

Case studies

This section tries to explore more deeply the disambiguation process for two particular topics, C145 and C164 in the case of German to Italian searches, in order to understand characteristics of each method. Table 5.9 shows scores of average precision and stems of each selected translation for topic C245, <TITLE>of which is "Japans Reisimport (Japan's rice import)". The decomposing algorithm used for German words broke automatically "Reisimport" into "Reis" and "Import", and transitive query translation was executed for the three source terms, "Japans", "Reis", and "Import" by using German to English and English to Italian dictionaries. In the process of translation, some irrelevant translations were obtained. For example, the Italian stem "viagg", which means "travel", was yielded from the German word "Reise", which was unfortunately converted into "Reis" by the stemming algorithm when the dictionary was implemented on the system. As shown in Table 5.9, performance of the best pair, cohesion and PRF based methods, was poor due to the fact that this irrelevant term was finally selected, whereas only the best sequence algorithm correctly chose stem "ris", which means "rice".

From Table 5.10 which indicates similarities of "ris" and "viagg" with the other translation

[6] Based on the result, the Dice coefficient was used in MLDC experiment discussed in Section 5.4.

Table 5.8: MAP scores in French to Italian and English to Italian (short query without feedback)

Methods		French to Italian	English to Italian
No disambiguation		0.1242	0.2093
1)Best pair	MI	0.1305	0.1889
	Dice	0.1957	0.2309
	Cosine	0.2225	0.2261
	Overlap	0.1822	0.2179
	Ave.	0.1827	0.2160
2)Best cohesion	MI	0.1668	0.2075
	Dice	0.1971	0.2290
	Cosine	0.2240	0.2215
	Overlap	0.1943	0.2112
	Ave.	0.1956	0.2173
3)Best sequence	MI	0.1751	0.2180
	Dice	0.1989	0.2296
	Cosine	0.2262	0.2221
	Overlap	0.2132	0.2140
	Ave.	0.2034	0.2209
4)PRF	Top 10	0.2260	0.2338
	Top 30	0.2273	0.2375
	Top 50	0.2286	0.2397
	Top 100	0.2314	0.2356
	Top 500	0.2215	0.2282
	Top 1000	0.2180	0.2216
	Ave.	0.2255	0.2327

Table 5.9: Average precision and selected translations for topic C145 (short query)

	Methods	Ave. Prec.	Translations (only stems)		
			Japans	Reis	Import
Cosine	Best pair	0.0714	giappon	viagg	import
Cosine	Best cohesion	0.0714	giappon	viagg	import
Cosine	Best sequence	1.0000	giappon	ris	import
PRF	Top 30	0.0714	giappon	viagg	import

Table 5.10: Cosine similarities in topic C145

Source terms	Translations (stems)	ris	viagg
Japans	giappon	0.00392	0.00593
	lacc	0.00000	0.00068
Import	import	0.00652	0.00370
	important	0.00160	0.00735
	introdurr	0.00000	0.00097
	rilev	0.00158	0.00330
	rilevant	0.00000	0.00197
	signific	0.00106	0.00241
	trascendent	0.00000	0.00126
	vistos	0.00000	0.00088

candidates, the reason why the best pair and cohesion algorithms consequently selected the irrelevant translation can be understood. As shown, many irrelevant translations were produced by the decomposing algorithm and transitive translation via English. Among them, "important" has the highest similarity with "viagg", and the best pair algorithm accidentally selected "viagg" for the source term "Reis" (fortunately, "important" was not adopted because "giappon" and "import" had already been chosen with higher similarity of 0.018). It is obvious from this case that the best pair algorithm has very limited ability for selecting correct translations due to locality in its selection process (i.e., only a single pair is considered in each step, independently), as discussed in Section 2.2.4.

For this case, unfortunately, the best cohesion algorithm also made an error. The total value of cohesion of "ris" is only 0.01044 (= "giappon" 0.00392 + "import" 0.00652), which is less than that of "viagg" ("giappon" 0.00593 + "important" 0.00735 = 0.01328). If the similarity between the pair of "viagg" and "giappon" are lower, then the cohesion method would correctly identify "ris". However, since the irrelevant translation "viagg" happens to have a close relationship with the other correct translation "giappon" (e.g., "travel to Japan"), the irrelevant term was selected by mistake. It should be noted that, although the best cohesion algorithm could not make a correct choice in this case, more information (i.e., similarity with "giappon") was used. So, its error probability is naturally expected to be lower than that of the best pair algorithm as indicated empirically in previous section.

In the best sequence algorithm, the similarity between "ris" and "import" has an effect on selection, because the cosine coefficient between "import" and "giappon" is very high as described above and consequently the combination of the three correct translations "import," "giappon" and "ris" has a high average score of similarities. Fortunately, the similarity between "import" and "viagg" is not so high, and therefore an irrelevant sequence of "import," "giappon" and "viagg" cannot be dominant. As shown in the case, the best sequence algorithm would keep the error probability lower than that of other methods by taking more evidence into account.

The PRF based method also erroneously selected the irrelevant translation "viagg" due to the fact that five documents in the top-ranked 30 documents included "viagg" whereas the number of documents containing "ris" was just two. However, in another topic C164, the PRF based method outperforms the best sequence algorithm. The short query of topic C164 is "Europäische Strafurteile zu Drogen (European sentences for drugs)". Table 5.11 shows average precision scores and selected translations (stems) for topic C164. The German word "Strafurteile" was decomposed into "Straf (penal)" and "Urteile (judgments)" by the algorithm used in this experiment and the best sequence algorithm chose an inappropriate translation "vol (i.e., volere)" for "Urteile". "Volere" is a polysemous word having more than one sense (i.e., "judge", "think", "maintain" and so on), and therefore the best sequence algorithm was inferior in performance to the PRF based method, which identified a more adequate translation "sentenz" which means "judgment", "sentence", "case", and so on, according to the fact that "sentenz" appears in 28 out of the top-ranked 30 documents (whereas "vol" occurs in only 8 documents).

The reason why "vol" was selected by the best sequence algorithm is that this term has a

Table 5.11: Average precision and selected translations for topic C145 (short query)

Methods		Ave. Prec.	Translations (only stems)			
			Europäische	Straf	Urteile	Drogen
Cosine	Best sequence	0.0648	europ	pun	vol	drog
PRF	Top 30	0.1187	europ	pun	sentenz	drog

Table 5.12: Cosine coefficients in topic C164

	europ	pun	drog	Total
vol	0.00935	0.00810	0.01983	0.03727
sentenz	0.00267	0.00880	0.02020	0.03168

relatively strong relationship with the correct translation "europ (Europe)" (see Table 5.12). As the average precision score in Table 5.11 and cosine similarities in Table 5.12 indicate, it cannot be definitively concluded that "vol" is a wrong translation, namely, the average precision score of the best sequence is not zero and "vol" has a relationship with "pun" and "drog" to some degree. From this viewpoint, the best sequence algorithm never makes a mistake, however, it does not allow to obtain the more appropriate translation "sentenz" due to the fact that a general term "europ" having a close relationship with the polysemous term "vol" happens to be included in the query. This would be a problem of using macro-statistics for translation selection.

Comparison of performance

Through the discussion on the two topics C145 and C164, it was clarified that success of the translation selection is strongly dependent on other translation candidates having a relationship with the correct translation. This is true not only for the term co-occurrence based method but also for the PRF based method. For example, in topic C145, the reason why this method erroneously selected "viagg" is perhaps because the list of translation candidates includes other Italian terms about "travel" such as "viaggiant (traveling)" and "viaggiator (traveler)" and therefore many documents on "travel to Japan" may accidentally be included in the top-ranked documents. The situational dependency may have made it harder for a statistically significant difference to be observed in previous section.

Thus it is difficult to draw a clear conclusion that be commonly applied to every case, and just some tentative findings on translation disambiguation were obtained in this experiment. For instance, the best sequence algorithm is likely to be dominant, followed in order by the best cohesion and the best pair algorithms, and it seems that the PRF based method often outperforms the best cohesion and pair algorithms. However, these tentative conclusions are an important starting point, and they may be used as a hypothesis for further experiments based on other test collections.

Efficiency of processing for IR

One problem for implementing the PRF based method would be that processing for it takes longer time in on-line IR settings because the document collection is repeatedly searched two times. If only a few translations are entered, then the search result would be obtained within a reasonable response time. However, in general, a set of translation candidates tends to contain many terms. Thus the disambiguation technique based on PRF would be appropriate for batch mode searching rather than usual on-line searching.

In contrast, it should be noted that no extra device is needed for executing the PRF based method other than the standard search function with document ranking and PRF, which is an advantage of this method for IR. In the case of the term co-occurrence based method, a special index file for quickly finding and extracting co-occurrence information is required. If the size of the document collection is large, then it may be difficult to prepare such an index file because too

many pairs of terms are included in the whole collection (huge computer resources may be required for implementation). Thus the situation in which translation disambiguation is needed should be carefully considered. This experiment suggested that there is no large difference in 'effectiveness' between the term co-occurrence and PRF based methods, and so efficiency may be a key factor when choosing a disambiguation method [7].

5.3 Experiment of Monolingual Document Clustering

This section reports an experiment on monolingual document clustering, which is partly based on Kishida(2010) [152].

5.3.1 Purpose

Before attempting an experiment on multilingual document clustering (MLDC), it is necessary to ascertain effectiveness of the leader-follower clustering (LFC) algorithm in comparison with other algorithms even though the LFC algorithm has an advantage in terms of efficiency. As discussed above, there are not so many algorithms appropriate to large-scale document clustering. Particularly, this experiment used two algorithms,

1. Spherical k-means (SKM) algorithm (batch mode)
2. Hierarchical Dirichlet process (HDP) mixture model

as baselines for articulating effectiveness of the LFC algorithm. Both the methods are based on iterative computation whereas the LFC algorithm can generate clusters after only single or double scans of the target document file, which means that the LFC algorithm clearly is superior to the SKM algorithm and the HDP mixture model in terms of efficiency. The purpose of the experiment is to examine empirically validity or quality of clusters generated by the LFC algorithm in a situation of 'monolingual' document clustering. By comparing the cluster validity with that achieved by the SKM and HDP methods, the effectiveness of the LFC algorithm will become clear. If 'cost-effectiveness' of the LFC is empirically confirmed, then MLDC based on the LFC algorithm would be able to obtain a reality.

The second purpose of the experiment on monolingual DC is to verify effectiveness of the LFC algorithm for large-scale document sets. Even though the LFC algorithm is well-known as an efficient clustering algorithm, whether the effectiveness is kept for large-size sets of documents or not has not yet been sufficiently clear [8]. In this experiment, the LFC algorithm was applied to a set of over 70,000 documents, and the cluster validity was examined for it.

5.3.2 Dataset and text processing

The Reuter corpus RCV1 created as a test collection for text categorization [187] was used to compare the effectiveness of document clustering techniques. Since one or more 'topic codes' are assigned to each record of the corpus, which can be used as 'answers' of clustering, the validity of clusters generated by each algorithm can be assessed based on the topic codes (note that the topic codes were used only for evaluation). Particularly, in this experiment, among news articles published during August 1996, a set of 6,374 records to which just a single topic code is assigned was extracted from the collection and used as a sample (i.e., $N = 6374$) because evaluation of clustering results including multi-topic documents becomes too complicated. In total, 68 different topic codes appear in the 6,374 records. The number of documents to which each topic code is assigned is shown in Figure 5.3, in which the 68 topic codes are sorted in descending order of the number of documents on the x-axis. As indicated in the figure, the distribution is somewhat

[7] As discussed later, because document translation is required for MLDC, the PRF based disambiguation is not appropriate. If the method is applied, then a set of translations for each sentence of a document is considered as a query and searching the target set for the long query would have to be repeated for all sentences (perhaps, it may be impossible to execute the PRF based method for a document vector).

[8] This problem was partly solved by Kishida(2010) [152], method and experiment of which will be described in this part.

skewed. Actual subject of each topic code and its number of documents is listed in Table 5.16 of Section 5.4.

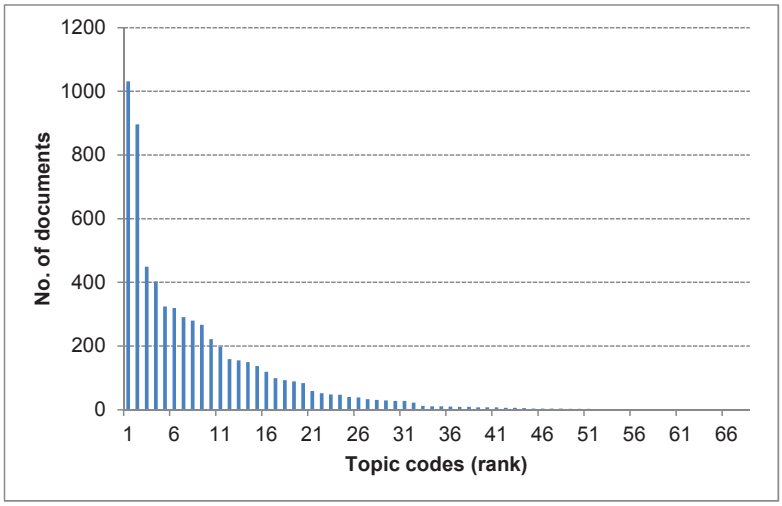


Figure 5.3: Rank distribution of topic codes in the collection

Similarly to the previous experiment, by standard text processing which consists of tokenization, removing stopwords and stemming by Porter's algorithm, document vectors for clustering were generated from the records. As discussed in Section 4.1.5, the idf factor (Section 2.1.2) can not be directly incorporated into probabilistic mixture model. So, in the first experiment, term frequency was simply used as elements of document vectors, and instead of incorporating the idf factor into the element, 'non-specific' terms appearing in more than 10% of all documents (i.e., over 637 documents) were removed from all document vectors. Also, terms appearing in only one document were not used as features for clustering. As a result, in total, 19,610 distinct terms appeared in the collection (i.e., $M = 19610$) and the average document length was 99.99.

The document set with 6,374 records and the vocabulary including 19,610 index terms were used for the first experiment of monolingual DC in which effectiveness of the LFC algorithm was compared with that of the SKM algorithm and the HDP mixture model. Meanwhile, in the second experiment, all index terms appearing just one document or over 10% of all documents are also counted unlike the first experiment because tf-idf weighting was used for elements of document vectors (of course, terms appearing in a single document does not contribute to computation of inner product). As a result, 44,770 distinct terms were extracted from 6,374 records and used as features (the indexing rule was very slightly different from that in the first experiment, but it does not affect interpretation of the experimental results).

5.3.3 Implementation of clustering methods

The document vector was defined by $\mathbf{d}_i = [\mathrm{f}_{i1}, \ldots, \mathrm{f}_{iM}]^T$ as mentioned above, and the element of cluster vectors was simply computed such that $\tilde{w}_{kj} = \sum_{i:d_i \in C_k} \mathrm{f}_{ij}$ for the LFC algorithm ($i = 1, \ldots, N$; $k = 1, \ldots, L$; $j = 1, \ldots, M$) in the first experiment.

For clustering by the HDP mixture model, Gibbs sampling based Chinese restaurant franchise (CRF) model was employed (Figure 4.15 in Section 4.1.5). As the SKM algorithm, this experiment used a modified Hartigan-Wong algorithm in which the inner product of unit vectors of a document and a cluster is computed and Equation (3.47) and (3.48) are incorporated for reallocation of documents (L documents at top positions in the file were automatically selected as seeds).

In contrast, the second experiment using larger datasets (see below) adopted standard tf-idf weighing scheme because only the LFC algorithm was applied. More specifically, an element of document vector $\mathbf{d}_i$ was defined by

$$w_{ij} = (\log \mathrm{f}_{ij} + 1.0) \times \log \frac{N}{n_j}, \tag{5.1}$$

Table 5.13: nMI scores of clustering results

Methods	Parameters	nMI score	L	No. of scans
Spherical k-means	$L = 80$	.4200	80	47
	$L = 74$	.4315	74	43
	$L = 68$	.4313	68	30
	$L = 54$	.4454	54	39
HDP mixture	$\alpha = 0.1, \beta = 0.01, \gamma = 0.1$	.4715*	54.1*	3000
	$\alpha = 0.1, \beta = 0.01, \gamma = 0.5$	.4725*	73.7*	3000
	$\alpha = 0.1, \beta = 0.01, \gamma = 1.0$	.4486*	84.3*	3000
LFC (double-pass)	$\theta_s = 0.04$	.3185	17	2
	$\theta_s = 0.05$	.4713	29	2
	$\theta_s = 0.06$	.4644	47	2
	$\theta_s = 0.07$	.4527	80	2
	$\theta_s = 0.08$	.4499	104	2
	$\theta_s = 0.09$	.4383	130	2
	$\theta_s = 0.10$	.4421	167	2
	$\theta_s = 0.11$	.4338	200	2
	$\theta_s = 0.12$	.3966	243	2

Note: *An average over 10 samples.

and that of cluster vector $\mathbf{c}_k$ was determined by

$$\tilde{w}_{kj} = \log \sum_{i:d_i \in C_k} \mathrm{f}_{ij} + 1.0. \tag{5.2}$$

When the target document set is very large, it may not be necessary to read all documents for generating cluster vectors in the first scan (see Section 3.3.2) [9]. In this experiment, only top 2,000 documents were checked in the first scan of the double-pass LFC algorithm, which is called 'modified double-pass method' for convenience in this thesis.

5.3.4 Results and analysis

Comparison of effectiveness

In this experiment, the cluster validity was measured by the normalized mutual information (nMI) in Equation (3.18) of Section 3.1.4 based on the topic codes of RCV1. The nMI scores of each clustering method are shown in Table 5.13.

As shown in the table, some sets of parameters were used in this experiment. In the HDP mixture model, three patterns of hyperparameters were examined. Among them, $\alpha = 0.1, \beta = 0.01$ and $\gamma = 0.5$ showed the highest nMI score. Because clustering by the HDP mixture model is based on the Gibbs sampling, a set of clusters can be obtained in each sample by allocating each document to the dish (i.e., topic) of a table at which the most tokens of the document are sitting (see Section 4.1.5). Therefore, in this experiment, after the Gibbs sampling was executed 3,000 times iteratively (note that this is a single chain), 10 samples extracted from the 2010th iteration to 3000th iteration with 10 intervals were used for generating sets of clusters, respectively, and the nMI score was averaged over the ten samples, which is indicated in Table 5.13. In the case that $\alpha = 0.1, \beta = 0.01$ and $\gamma = 0.5$, the average was 0.4725, which outperforms those by the SKM and the LFC algorithms within this table [10].

In the LFC algorithm, threshold value θ_s for merging a document into a cluster has to be determined a priori (Section 3.3.2). In the case that $\theta_s = 0.05$, the highest nMI score (= .4713)

[9] This idea was proposed by Kishida(2010) [152].

[10] However, it may be possible that 3,000 iterations were insufficient for attaining enough convergence. Because more than 3,000 iterations for 6,374 documents was considered as unrealistic and the HDP mixture model is not the main target of this thesis, the problem of convergence when applying the HDP mixture model to document clustering is left for future research.

was obtained in this experiment. However, the number of clusters is only 29 (the number of topic codes is 68), and the threshold may be inappropriate in this sense. Actually, by trial and error, it turned out that threshold $\theta_s = 0.06571$ achieves 0.4770 of the nMI score [11], which is slightly superior to the highest nMI score in the HDP mixture model (i.e., 0.4747 at 2930th sample when $\alpha = 0.1, \beta = 0.01$ and $\gamma = 0.5$). Inevitably, the result does not mean dominance of the LFC algorithm because other sets of hyperparameters in the HDP mixture model were not examined.

Rather, an important thing is that the LFC algorithm generated clusters with quality 'comparable' to those obtained by the HDP mixture model. The confusion matrix, which is a cross tabulation of the number of documents by topic code and generated cluster (see Table 3.5), is shown in Figures 5.4 and 5.5, respectively, as a two-dimensional histogram. In these graphs, clusters are ordered by the number of documents belonging to them, and topic codes are arranged in alphabetical order. Although it is hard to understand its details due to the fact that there are many bars in the graph, it does not seem that 'anomalous' or 'erroneous' results were obtained by the two clustering executions.

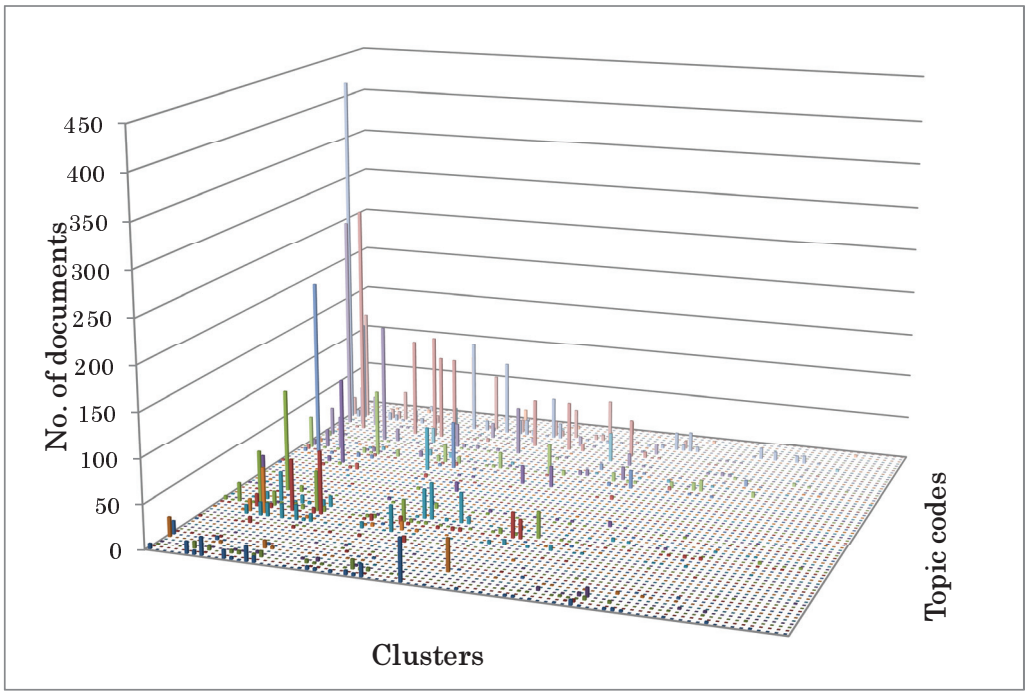


Figure 5.4: Confusion matrix (1): LFC algorithm ($\theta_s = 0.07$)

Unfortunately, it seems that the SKM algorithm provided slightly 'poor' clustering results in this experiment. In the case of the k-means algorithm, the number of clusters has to be provided a priori (see Section 3.3.1), and this experiment attempted four cases, $L = 54, 68, 74, 80$. Among them, 54 and 74 were adopted from results of the HDP mixture model, and $L = 80$ was used for comparing with the LFC algorithm when $\theta_s = 0.07$. Needless to say, 68 is the number of topic codes. The nMI scores were relatively lower than those by the HDP mixture model and the LFC algorithm. Note that this result does not necessarily mean inferiority of k-means algorithms again because there are other k-means algorithms and the difference of nMI scores was small.

Table 5.13 indicates the number of times that the document file was scanned by each algorithm. In the case of the SKM based on the Hartigan-Wong algorithm, scans from 30 to 50 were needed until the results converged, which is about ten times more than that in the LFC algorithm. This means that computational complexity of the LFC algorithm is actually lower than that of the SKM algorithm even though the LFC algorithm provides clustering results with quality comparable to that by the HDP mixture model and the SKM algorithm based on iterative computation. Therefore, in terms of 'cost-effectiveness', it can be concluded that the LFC algorithm clearly outperforms the HDP mixture model and the SKM algorithm.

[11] The number of generated clusters was 68, which is equal to the number of topic codes.

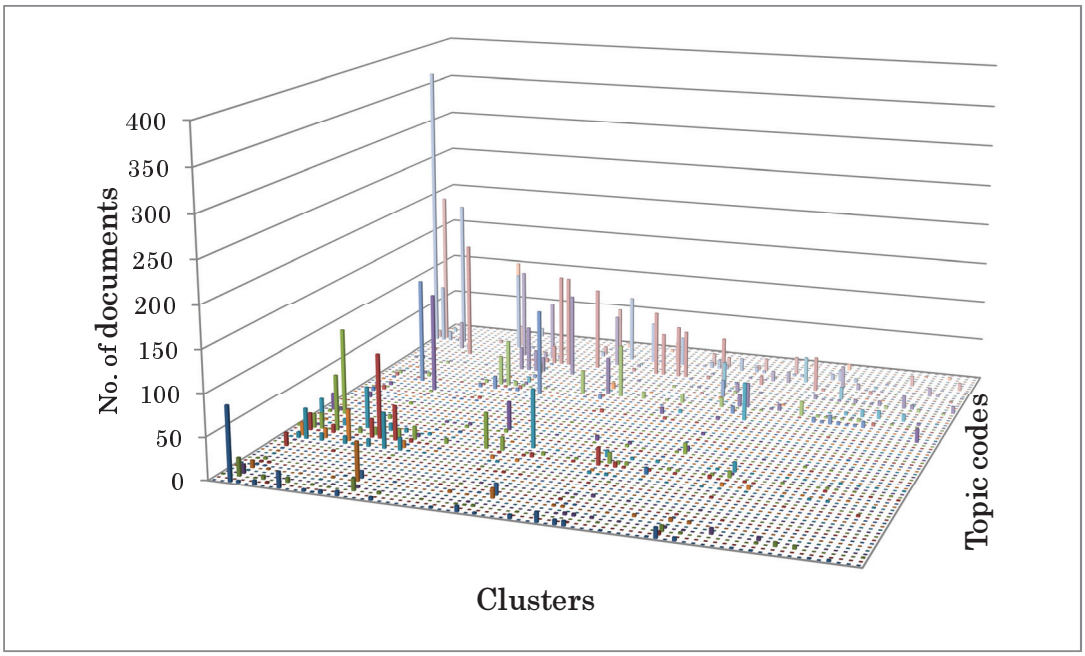


Figure 5.5: Confusion matrix (2): HDP mixture ($\alpha = 0.1, \beta = 0.01, \gamma = 0.5$, 2930th sample)

Effectiveness for larger datasets

In the second part of this experiment, effectiveness of the LFC algorithm was examined for larger document sets. In terms of executing DC, the set with 6,374 records would be enough large, but in real situations, larger document collections have to be processed. For the experiment, more records with a single topic code were extracted and added to the document collection as shown in Table 5.14.

For example, the "Aug.-Sep. 1996" dataset consists of 22,892 documents extracted from the RCV1 corpus in the period of August to September 1996. This dataset was created by adding news articles in September 1996 to the original test dataset used in the first experiment, and its size was increased about 3.4 times. Similarly, as a result of adding news articles cumulatively, the "Aug.-Oct." dataset included 40,697 documents (about 6.4 times). Also, the numbers of documents in the "Aug.-Nov." and "Aug.-Dec." datasets amount to 57,076 (about 9.0 times) and 72,835 (about 11.5 times), respectively. It should be noted that only documents having a single topic code were included in the datasets according to the same procedure in the first experiment, and the same text processing was executed on the datasets. Note that only $\theta_s = 0.08$ was used as the threshold in the second experiment of this section.

As indicated in Table 5.14, the modified double-pass algorithm worked well for the larger dataset. For instance, 77 clusters were generated for 79 topic codes and nMI score was relatively high (=0.462) in the "Aug.-Sep." dataset. The same tendency can be observed even if documents included in the dataset more increase. The nMI scores were 0.459 for 40,697 documents of the "Aug.-Oct." dataset, 0.462 for 57,076 documents of the "Aug.-Nov." dataset, and 0.453 for 72,835 documents of the "Aug.-Dec." dataset (see Table 5.14). This result indicates a possibility that the modified double-pass algorithm can provide effective clustering results for larger datasets.

The modified double-pass algorithm in which only top n documents are checked in the first scan (i.e., $n < N$) is always executed in less computational complexity than the original double-pass algorithm, and its effectiveness of clustering was also improved in this experiment (results by the original algorithm are shown as 'baseline' in Table 5.14) [12]. If there are enough representative documents up to n documents in the file, then it is expected that this algorithm works effectively. This condition is considered to be satisfied in the sample used for this experiment. Figure 5.3 indicates that the test set contains relatively few large 'true' clusters and many small ones. In such

[12] Of course, by looking for optimal values of θ_s more carefully, the original algorithm may provide better results. It is enough to confirm the effectiveness of the modified double-pass algorithm here.

Table 5.14: Performance of LFC algorithm for larger dataset

	Dataset: Year - 1996			
	Aug. - Sep.	Aug.-Oct.	Aug.-Nov.	Aug.-Dec.
No. of documents	22892	40697	57076	72835
No. of topic codes	79	82	83	86
Baseline: Original double-pass method by Crouch(1975) [71]				
No. of clusters	180	250	305	354
nMI	0.417	0.388	0.375	0.365
Elapsed time (sec.)	6357.5	19629.3	37996.9	61336.5
Modified double-pass method (2000 docs) by Kishida(2010) [152]				
No. of clusters	77	84	85	85
nMI	0.462	0.459	0.462	0.453
(% of the baseline)	110.8%	118.3%	123.2%	124.1%
Elapsed time (sec.)	3445.1	10091.4	19356.1	31292.2
(% of the baseline)	54.2%	51.4%	50.9%	51.0%

Note: $\theta_s = 0.08$.

skewed distribution, documents belonging to large 'true' clusters tend to be included in processing of the first scan, and so, cluster vectors generated by these documents are likely to work positively for generating homogeneous clusters related to the large 'true' clusters in the second scan, which would increase the quality of the clustering result.

Actually, not all topic codes appear up to 2,000 documents in the file as shown in Figure 5.6, which is a plot of cumulative numbers of distinct topic codes appearing in each position from the top of the test file with 6,374 records. Nevertheless, the modified double-pass algorithm effectively returned clustering results with high quality in this experiment, and the skewness in the set of 'true' clusters would enable the modified double-pass algorithm to provide good clustering results.

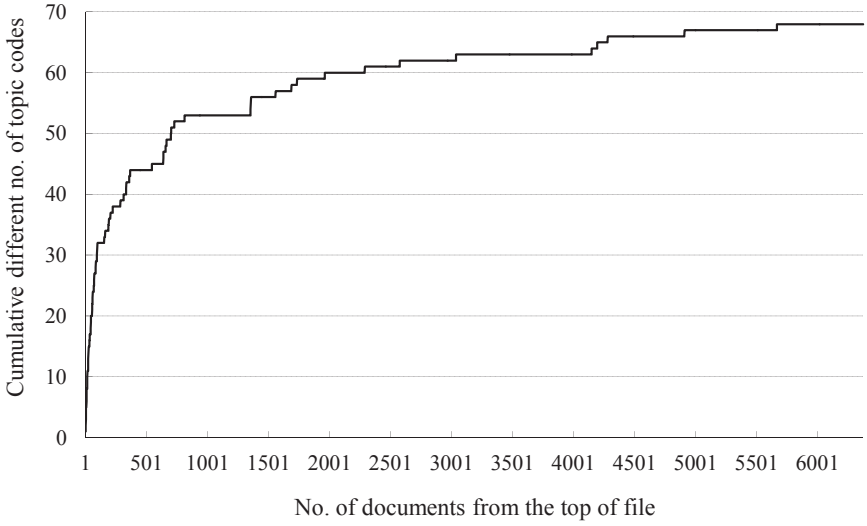


Figure 5.6: Cumulative numbers of distinct topic codes appearing in each position of the sample file

Conversely, it is not guaranteed that the modified double-pass algorithm works well on a document set that does not have a skewed distribution of documents over 'true' clusters unlike the test data in this experiment. The ideal situation would be that representative documents of all 'true' clusters appear up to n documents from the top of the file, and the distribution of documents over 'true' clusters in the set of n documents is similar to that in the entire set of documents. In such situation, the modified double-pass algorithm would be able to return a clustering result without

creating insufficient cluster vectors.

5.4 Experiment of Multilingual Document Clustering

This section reports an experiment of large-scale multilingual document clustering (MLDC) based on Kishida(2011) [154], which is the main topic of this thesis. Research findings in previous sections were utilized for designing the experiment discussed in this section.

5.4.1 Purpose

More specifically, the experiment has three purposes:

1. To verify the degree of effectiveness of MLDC compared to monolingual DC.
2. To compare the effectiveness of MLDC between two strategies of cluster translation and document translation.
3. To confirm whether translation disambiguation improves the effectiveness of MLDC or not.

Cluster translation and document translation will be explained later.

5.4.2 Clustering strategy for multilingual documents

Dictionary-based translation

As mentioned above, simple dictionary-based translation is adopted for MLDC in this thesis. Generally, bilingual dictionaries are easy to obtain, and especially, dictionaries for translation between each language and English are more readily available because English is the prevailing international language. Thus even if the target set consists of documents written in various languages, automatically converting words in the documents into English equivalents by using the machine-readable bilingual dictionaries allows to execute a clustering algorithm for the multilingual set. Also, since the conversion of words is a simple operation with searching dictionaries, it is easy to implement the module into a DC system. From these reasons, dictionary-based translation should be closely examined as a first step for developing effective and efficient algorithms of MLDC.

In CLIR studies, some researchers have adopted a strategy of translating target documents, not search queries, using bilingual dictionaries or bilingual term lists (see Section 2.2.2). Basically, this thesis attempts to apply the strategy to the MLDC problem.

Document translation vs. cluster translation

It is possible to translate clusters after executing monolingual clustering independently for respective parts of each language, rather than directly translating individual documents before the clustering operation. Specifically, terms in representation of each cluster generated by the monolingual clustering at the first stage are converted into English ones by the dictionary-based translation, and then a clustering algorithm is again applied to the set of translated clusters at the second stage for outputting a final result of MLDC (see the left side of Figure 5.7). If 'correct' homogeneous clusters are created at the first stage and translation of clusters is successful, then this 'cluster translation' strategy would show good performance. When documents are grouped into a sufficiently small number of clusters at the first stage, the processing time for cluster translation is expected to be shorter than for document translation in which all documents have to be translated (see the right side of Figure 5.7) even though it is necessary to repeat the clustering operation twice. This is an advantage of cluster translation.

Translation disambiguation

For a source word, an entry in a bilingual dictionary often provides multiple translations with different senses, and therefore, word sense disambiguation for the translations (i.e., translation disambiguation) is important for executing CLIR (Section 2.2.4).

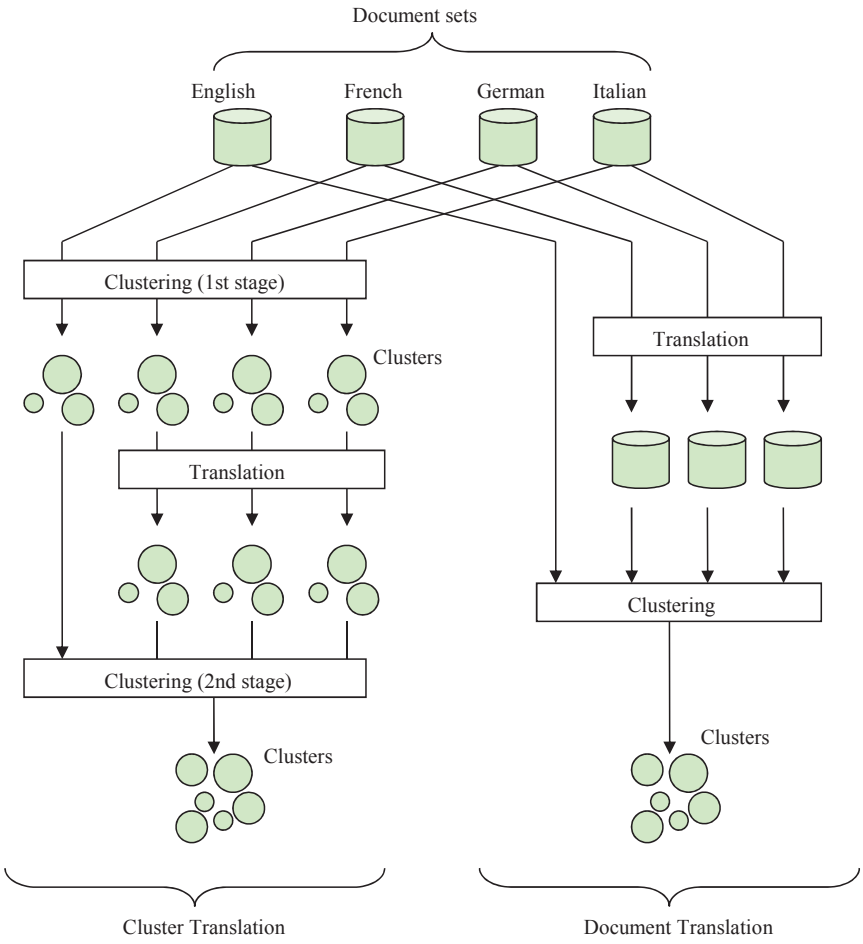


Figure 5.7: Cluster translation and document translation for MLDC

Indeed, retrieval experiments have shown that translation disambiguation often improves search performance as indicated actually by the experiment in Section 5.2, so it is worth applying a disambiguation technique also to dictionary-based translation in MLDC. A variety of methods of translation disambiguation such as techniques using parallel corpora, techniques based on term co-occurrence statistics, techniques based on query expansion (QE) and so on, have been developed in the field of CLIR (Section 2.2.4). Among them, parallel corpus-based techniques would be inappropriate in situations that a pair of languages for which a parallel corpus is difficult to obtain is included in the target document set. Also, because QE-based techniques require IR operations on the target document set, it would be unrealistic to apply them to a large-scale multilingual document set. By regarding each document as a long query and searching the target document collection (i.e., the English document set) for the query, QE-based methods may be applied to MLDC, but the repeated searches for such long queries take too long time. Therefore, techniques based on term co-occurrence frequencies would be the most promising approach for disambiguation in practical MLDC. In particular, a version of this technique adopted by CLIR experiments in Adriani(2000) [1] and Gao et al.(2001) [101] was selected for this experiment because experiments in Section 5.2 showed that the method was sufficiently effective and its computational complexity is relatively low. More precisely, the 'best cohesion' method based on Equation (2.60) was employed in this experiment.

Double-pass algorithm for DC

This experiment used the double-pass leader-follower clustering (LFC) algorithm proposed by Crouch(1975) [71] for clustering documents or for grouping clusters according to experimental results in previous section. As discussed in 3.3.2, the target document file is scanned twice, and cluster vectors are generated at the first scan and each document is allocated to a cluster at the second scan in the double-pass algorithm. Although double-pass clustering takes longer time than single-pass clustering in which cluster generation and allocation are concurrently executed, the double-pass algorithm is known to provide better results. Because the difference in processing time between the two algorithms is not so large, this experiment adopted the double-pass algorithm. Also, the modified double-pass clustering [152] keeps high effectiveness for large-scale document collections as shown in previous section, but in this experiment, the modified version was not used (of course, the modified double-pass algorithm can be applied to MLDC situations).

Note that term weights in document and cluster vectors were computed according to Equations (5.1) and (5.2), respectively, for the double-pass LFC algorithm.

5.4.3 Dataset and text processing

Dataset for the experiment of MLDC was created by extracting a subset of records from the Reuters corpus RCV2, which is a multilingual document set developed for exploring text categorization [187]. The RCV2 contains a huge number of documents, and therefore it was impossible to employ all the documents in the experiment because the clustering operation had to be repeated many times with varying values of parameters in order to closely examine the clustering performance. A subset of the records was therefore selected based on the following rules:

1. Only four languages, English, French, German and Italian, are used because text processing for these four languages is relatively easy.

2. Only documents to which a single topic code is assigned are used because it is complicated to measure the effectiveness of clustering documents with two or more subject topics.

3. At least one thousand documents must be included in the subset of each language in order to keep a sufficient volume of test data.

Accordingly, data extraction was started from a subset of news articles published in August 1996, and proceeded chronologically to the next month if over 1,000 documents were not obtained in the month. While the August 1996 subset included over 1,000 English documents (in total, 6,374 documents as before), French and German documents were extracted from two subsets of August and September 1996 (2,403 and 3,153 documents, respectively). Partly because there are fewer

Table 5.15: Basic statistics of document sets for MLDC experiment

	No. of records	No. of different topic codes	No. of different index terms	Average length of document
English	6374	68	44758	193.015
French	2403	23	16896	166.128
German	3153	24	28924	177.513
Italian	1220	18	5929	76.642
Total	13150	69	96507	173.588

documents in Italian than in other languages, five subsets from August to December 1996 were checked and 1,220 documents were extracted. The numbers of documents in the experiment are summarized in Table 5.15, which indicates that the multilingual test set consists of 13,150 documents in total. Also, Table 5.15 shows the numbers of different topic codes appearing in each language subset (see Table 5.16 for details of the topic codes). These topic codes are considered to represent 'true' clusters in the dataset, namely, the experimental dataset contains 69 'true' clusters which were employed as correct answers for evaluating MLDC techniques. Note that the topic codes were used only for the evaluation and not for the clustering stage similarly to the experiment in previous section, because this thesis focuses on only unsupervised classification techniques.

Although a set of 13,150 documents is not large from a viewpoint of IR applications, its size was considered to be appropriate for repeated executions of clustering that is a time consuming task. It is expected that experimental results obtained from this medium-scale document set can be valid in situations of clustering larger collections unless homogeneity of documents is largely changed. For example, if only 69 topic codes shown in Table 5.15 are appearing in another larger subset of RCV2 and heterogeneous documents are not included in it, then clustering results would be almost identical. Actually, the previous section reported that clustering effectiveness did not largely change when the test set was enlarged from 6,374 documents to 72,835 documents, by using the RCV1 corpus (see Table 5.14).

Each term postulated as an analytical unit in calculation for clustering (i.e., Equations (5.1) and (5.2)) was automatically identified from text in each news article based on a standard indexing technique (Section 2.1.6), in which removal of stopwords and stemming of words were executed using linguistic resources (more precisely, stopword lists and Porter's stemmers for English, French, German, and Italian languages) developed by the Snowball Project [13]. For only German words, a simple heuristic was used to decompose compound words during the indexing process similarly to the CLIR experiment in Section 5.2. German strings surrounded by delimiters (e.g., blank, punctuation marks and so on) in the text were extracted and broken down into a set of German entries registered in a German dictionary. As this dictionary, a German-English bilingual dictionary discussed later was employed. For example, the German string, "Zeitungsberichten" can be decomposed into "Zeitung (newspaper)" and "berichten (inform)" by performing operations of matching entry terms of the dictionary if these two component words are listed as entries. More precisely, only the longest matching entries were used as index terms after stemming. For example, "Zeit (time)" was not extracted from "Zeitungsberichten" because "Zeitung" is longer than "Zeit" (namely, no string was extracted any more from the longest matching entry "Zeitung"). Also, an unknown substring not matching any entry was treated as an independent word if the substring contained three or more characters.

The heuristics may not provide always a linguistically correct solution, but it was expected to work well in the experiment for examining the performance of MLDC, in which the objective of decomposition is not to translate correctly each sentence, but to just enhance correctness of similarity measure between documents. Table 5.15 indicates the different number of index terms by each language, namely, 44,758 index terms appeared in English documents [14], 16,896 in French ones, 28,924 in German ones, and 5,929 in Italian ones. The average length of documents is

[13] http://snowball.tartarus.org//

[14] For some reasons, this number is slightly different from that in the second experiment reported by the previous section.

Table 5.16: Numbers of records in dataset by code and language

No	Code		English	French	German	Italian	Total
1	C11	Strategy/Plans	198	2		1	201
2	C12	Legal/Judicial	8				8
3	C13	Regulation/Policy	84	1		6	91
4	C14	Share Listings	40			6	46
5	C15	Performance	2				2
6	C151	Accounts/Earnings	93				93
7	C152	Comment/Forecasts	33				33
8	C16	Insolvency/Liquidity	9	1	6	1	17
9	C17	Funding/Capital	6		1	1	8
10	C171	Share Capital	8				8
11	C172	Bonds/Debt Issues	1				1
12	C173	Loans/Credits	2				2
13	C174	Credit Ratings	1				1
14	C181	Mergers/Acquisitions	31				31
15	C182	Asset Transfers	3				3
16	C183	Privatisations	1				1
17	C21	Production/Services	159				159
18	C22	New Products/Services	99				99
19	C23	Research/Development	11				11
20	C24	Capacity/Facilities	267	1			268
21	C31	Markets/Marketing	280				280
22	C32	Advertising/Promotion	10				10
23	C33	Contracts/Orders	222		2		224
24	C34	Monopolies/Competition	4				4
25	C41	Management	4		1		5
26	C42	Labour	2				2
27	E11	Economic Performance	150	145	80	97	472
28	E12	Monetary/Economic	89	117	142	144	492
29	E121	Money Supply	1				1
30	E131	Consumer Prices	2				2
31	E14	Consumer Finance	2				2
32	E143	Retail Sales	1				1
33	E21	Government Finance	4				4
34	E31	Output/Capacity	11				11
35	E311	Industrial Production	1				1
36	E411	Unemployment	1				1
37	E51	Trade/Reserves	9	24	7		40
38	E512	Merchandise Trade	28				28
39	E61	Housing Starts	4				4
40	E71	Leading Indicators	155				155
41	G15	European Community	52				52
42	G154	EC Monetary/Economic	1				1
43	GCRIM	Crime, Law Enforcement	320	374	608	70	1372
44	GDEF	Defence	29	78	140	10	257
45	GDIP	International Relations	291	456	439	33	1219
46	GDIS	Disasters And Accidents	119				119
47	GEDU	Education		10	14		24
48	GENT	Arts, Culture, Entertainment	48	74	44	3	169
49	GENV	Environment And Natural World	22				22
50	GHEA	Health	28	20	83		131
51	GIOB	Labour Issues	1	62	68	86	217
52	GODD	Human Interest	47	75	65		187
53	GPOL	Domestic Politics	324	401	639	216	1580
54	GPRO	Biographies, Personalities, People	59				59
55	GREL	Religion	3	36	24		63
56	GSCI	Science and Technology	8	27	30		65
57	GSPO	Sports	896				896
58	GTOUR	Travel and Tourism	2	3	4	2	11
59	GVIO	War, Civil War	403				403
60	GWEA	Weather	39				39
61	GWELF	Welfare, Social Services	2	25	19	12	58
62	M11	Equity Markets	1031	283	30	172	1516
63	M12	Bond Markets	449	187	703	357	1696
64	M13	Money Markets	1	1	2	3	7
65	M131	Interbank Markets	3				3
66	M132	Forex Markets	5				5
67	M14	Commodity Markets	137	2			139
68	M141	Soft Commodities	12				12
69	M142	Metals Trading	6				6
		Total	6374	2403	3153	1220	13150

also indicated in Table 5.15, which shows that English news articles were the longest on average (average document length of 193.02), followed by German and French ones. Among the four languages, Italian documents were the shortest, with an average length of less than half of that of English ones.

5.4.4 Implementation of clustering and translating processes

Clustering process

In this experiment, the double-pass algorithm based on Equations (5.1) and (5.2) was used in all operations of clustering. The values of N and n_j in the equations were estimated for respective parts of each language (e.g., $N = 2403$ for French documents), and weight w_{ij} of each source term was used for the corresponding translations with no change in the case of MLDC. Also, as heuristics, all documents for which the maximum similarity was lower than 0.01 were forcibly allocated to a 'miscellaneous' cluster in the second scan of the double-pass algorithm. When grouping clusters at the second stage in the cluster translation strategy, the same procedure can be applied by regarding a monolingual cluster as a single document. Therefore, Equation (5.1) was used for the monolingual cluster and Equation (5.2) for multilingual clusters generated at the second stage of the cluster translation strategy.

Translating process

In this experiment, simple dictionary-based translation to English was executed in both the strategies of document translation and cluster translation, namely, French, German and Italian index terms identified by the indexing process were simply replaced by the corresponding English words listed in entries of machine-readable bilingual dictionaries. French to English, German to English and Italian to English dictionaries were downloaded from the web site of freelang.com in January 2010 [15] and used for the replacement. The French to English dictionary contains over 36,000 French entries and 1.28 translations per entry are listed (the standard deviation is about 0.64). Also, the German to English dictionary includes over 173,000 entries (1.05 translations per entry and the standard deviation is about 0.28) and the Italian to English dictionary has over 39,000 entries (1.53 translations per entry and the standard deviation is about 0.99). On average, entries of the Italian to English dictionary list more English terms than any other dictionaries.

All corresponding English words were stemmed by Porter's algorithm and recorded as index terms for MLDC. In order to avoid incomplete identification of relevant entries, all headwords of entries were stemmed before searching the dictionaries (see Section 2.2.3 for details of stemming in the matching operation). Nevertheless, if no entry matching French, German and Italian words were found in the bilingual dictionaries, then these words were sent to Porter's English stemmer with no change and were straightforwardly treated as English words for MLDC because they may be proper nouns such as names of places, persons, and so on [16].

In the case of disambiguating translations, the technique based on term co-occurrence frequencies was directly applied to the sets of English words obtained from the bilingual dictionaries after stemming. However, in the case of document translation, complexity of computing 'cohesion' $C(t, T_k)$ in Equation (2.60) is considerably high because many words are included in each document or cluster vector, unlike query translation in standard CLIR. One of the solutions is to extract a fixed number of index terms from each document or cluster, which corresponds to feature selection in text categorization techniques (Section 3.1.3). This experiment adopted 10, 30, 50 and 100 index terms selected from each document or cluster, which means that $m = 10, 30, 50, 100$ in Equation (2.60), and top-ranked terms which were ordered based on weight in Equation (5.1) were chosen as the features. Also, when no disambiguation technique was applied, the set of English words obtained from the bilingual dictionaries was straightforwardly adopted as the features of document or cluster vectors after stemming, in which 10, 30, 50 and 100 words were selected in the

[15] http://www.freelang.com/. The dictionaries were used in the CLIR experiment of Section 5.2, but the versions are different.

[16] Note that the translation process is almost same with that in the CLIR experiment of previous section.

Table 5.17: Results of clustering by each language (monolingual DC)

	English		French		German		Italian		Total no.
θ_s	L	nMI	L	nMI	L	nMI	L	nMI	of clusters
0.04	9	0.327	4	0.301	3	0.273	3	0.236	19
0.06	29	0.474	11	0.355	16	0.482	7	0.461	63
0.08	91	0.477	41	0.477	45	0.444	8	0.508	185
0.10	218	0.453	114	0.423	111	0.417	21	0.548	464
0.12	404	0.438	217	0.403	201	0.383	45	0.528	867
0.14	639	0.438	309	0.401	318	0.379	69	0.463	1335
Ave.	231.7	0.435	116.0	0.393	115.7	0.396	25.5	0.457	-

same manner for comparison. As similarity measure $\text{sim}(t, t')$ in Equation (2.59), this experiment employed the Dice coefficient [17].

5.4.5 Results and analysis

Monolingual clustering by each language

Before examining MLDC techniques, performance level of monolingual clustering for each language should be confirmed. Table 5.17 shows the numbers of generated clusters and nMI scores in Equation (3.18) as a result of monolingual clustering by each language, in which a set of values of threshold θ_s for merging, {0.04, 0.06, 0.08, 0.10, 0.12, 0.14}, was adopted. Because it is difficult to determine an optimal threshold in the LFC algorithm, this combination of threshold values was empirically selected in this experiment for checking overall trend of the effectiveness of clustering. For example, 0.477 at the threshold of 0.08 is the maximum nMI score of English DC among the scores in Table 5.17 [18], but it is not guaranteed that this value represents the true highest effectiveness. This fact suggests at most that the maximum score would not be significantly larger than 0.477 and that the highest effectiveness may be obtained in a threshold near 0.08 (it can not be completely rejected that 0.477 is the maximum score). The purpose of this experiment is not to find precisely the maximum nMI score, but to observe the overall trend, and then such kind of sets of threshold values will be used in successive analyses in this experiment. Regarding other languages, the maximum values of nMI in Table 5.17 were 0.477 in French (with threshold of 0.08), 0.482 in German (with threshold of 0.06), and 0.548 in Italian (with threshold of 0.10). On average, Italian clustering was the most successful, followed by English (i.e., the average nMI score over all threshold values is 0.457 in Italian and 0.435 in English). In the situation that an optimal threshold is unknown, it can be interpreted that higher average of nMI scores over thresholds indicate more successful clustering of documents.

The clustering performance in German and French sets of documents was slightly lower but sufficiently comparable to that of English clustering (i.e., the average was 0.393 in French and 0.396 in German). This means that the text processing procedure in this experiment worked almost equally well in all four languages (at least, there would be no serious problem for it).

Cluster translation

In the cluster translation strategy, sets of clusters generated by the double-pass LFC algorithm for each language were translated into English, and were again merged into some groups by the same procedure of the double-pass technique (see Figure 5.7). As discussed above, the translation can be executed either without any disambiguation or with disambiguation. The effectiveness of MLDC based on cluster translation without disambiguation is shown in Table 5.18. The number of index terms entered into the translation process was limited to four cases, namely, (a) 10, (b) 30, (c) 50 and (d) 100 terms, and the set of threshold values for DC at the first stage (i.e., monolingual DC

[17] According to an experimental result in Section 5.2, there is almost no difference of effectiveness between the Dice and cosine coefficients for translation disambiguation.

[18] Inevitably, the score is not largely different from that by monolingual DC in the experiment of previous section.

Table 5.18: Results of MLDC by cluster translation without disambiguation

Thresholds at 2nd stage	Thresholds at 1st stage (cluster generation) 0.06 L	0.06 nMI	0.08 L	0.08 nMI	0.10 L	0.10 nMI	0.12 L	0.12 nMI
(a) No. of selected terms = 10								
0.06	47	0.341	124	0.342	218	0.317	323	0.302
0.08	49	0.346	137	0.349	258	0.330	397	0.316
0.10	55	0.352	155	0.358	307	0.341	478	0.327
0.12	60	0.357	164	0.361	346	0.345	561	0.336
Ave.	52.8	0.349	145.0	0.352	282.3	0.334	439.8	0.320
(b) No. of selected terms = 30								
0.06	52	0.338	127	0.342	218	0.313	289	0.292
0.08	58	0.347	149	0.354	288	0.329	420	0.315
0.10	60	0.351	161	0.364	337	0.339	528	0.327
0.12	62	0.358	170	0.367	386	0.350	619	0.336
Ave.	58.0	0.348	151.8	0.357	307.3	0.333	464.0	0.318
(c) No. of selected terms = 50								
0.06	52	0.346	126	0.343	192	0.300	232	0.281
0.08	56	0.355	151	0.357	271	0.321	374	0.306
0.10	60	0.354	163	0.365	331	0.332	495	0.320
0.12	62	0.358	172	0.371	373	0.339	597	0.335
Ave.	57.5	0.353	153.0	0.359	291.8	0.323	424.5	0.310
(d) No. of selected terms = 100								
0.06	49	0.344	94	0.316	128	0.276	118	0.247
0.08	54	0.351	131	0.347	206	0.301	216	0.267
0.10	62	0.358	154	0.364	281	0.323	362	0.294
0.12	62	0.358	165	0.367	338	0.336	499	0.316
Ave.	56.8	0.353	136.0	0.349	238.3	0.309	298.8	0.281

Note: Term selection was executed only at second stage.

by each language) was reduced to {0.06, 0.08, 0.10, 0.12} by removing 0.04 and 0.14 because in the case of 0.04 and 0.14, their effectiveness was relatively lower (see Table 5.17).

The nMI scores listed in Table 5.18 clearly show that even if many more terms in cluster vectors were translated, the effectiveness of final MLDC was not always improved. For example, when the threshold at the first stage is 0.12, the average scores of nMI indicated in Table 5.18 decrease as the number of selected terms becomes large, namely, 0.320 for 10 terms, 0.318 for 30 terms, 0.310 for 50 terms, and 0.281 for 100 terms. In the case of a threshold of 0.10, a similar monotonically decreasing pattern was found, but when the threshold was 0.06 and 0.08, no consistent increasing or decreasing pattern was observed.

Table 5.19 shows the numbers of translations entered into clustering operations at the second stage for merging clusters generated at the first stage. Because there are actually source terms having two or more translations, clustering at the second stage was executed based on many more translated English terms than the number of selected (source) terms in the case of French, German and Italian clusters if translation is executed with no disambiguation. For example, when the number of selected terms was 10 and the threshold was 0.06, 10.64 English translations were generated from French clusters on average (per cluster).

A similar conclusion can be obtained for cluster translation with disambiguation, result of which is shown in Table 5.20. Only in the case of the threshold of 0.08, a monotonically increasing pattern was observed, while for the other values of threshold, no consistent increasing or decreasing pattern was found. In order to articulate this observation more clearly, two-way ANOVA with repeated measurement was attempted for the two factors of disambiguation (its levels are 'with' and 'without') and the number of selected terms (its levels are '10', '30', '50' and '100'). Note that data for the analysis are all nMI scores enumerated in Tables 5.18 and 5.20 (except for average

Table 5.19: Statistics on numbers of translations in cluster translation strategy without disambiguation

	French		German		Italian	
θ_s	Mean	Std.Dev.	Mean	Std.Dev.	Mean	Std.Dev.
(a) No. of selected terms = 10						
0.06	10.64	3.05	9.81	3.26	18.29	6.54
0.08	11.56	4.41	11.87	5.00	18.00	5.50
0.10	11.48	3.84	12.45	5.04	24.24	8.78
0.12	11.97	4.00	12.92	5.09	28.27	10.96
(b) No. of selected terms = 30						
0.06	33.00	8.41	30.75	5.23	72.43	24.77
0.08	36.90	9.79	40.24	14.77	75.63	20.48
0.10	39.04	9.72	44.41	14.66	85.76	30.57
0.12	39.94	9.38	46.86	15.86	96.00	27.91
(c) No. of selected terms = 50						
0.06	54.00	10.72	53.00	7.39	132.00	37.35
0.08	59.66	15.20	71.73	25.74	137.50	29.99
0.10	64.37	15.45	82.67	28.65	135.71	43.43
0.12	67.26	15.14	89.33	28.56	158.87	46.41
(c) No. of selected terms = 100						
0.06	105.09	24.83	126.06	22.75	274.71	42.68
0.08	103.83	34.09	144.22	37.00	268.75	39.19
0.10	110.09	35.14	167.11	46.67	265.57	103.17
0.12	115.45	36.00	177.30	47.67	288.11	108.32

scores). Although it is not possible to conduct an exact statistical test in this experiment, ANOVA would be useful to grasp an overall tendency of clustering effectiveness under the situation that an optimal threshold can not be determined a priori. The results of ANOVA are shown in Table 5.21, implying that both the factors of the disambiguation and the number of selected terms have no statistically significant effect on clustering effectiveness (i.e., the p-values are 0.343 and 0.173, respectively), and also, the p-value of the cross-effect was 0.636, which means that the null-hypothesis can not be rejected at the significance level of 5% for all factors. However, the p-value of the number of selected terms is relatively small (p = 0.173), and so it may be difficult to reject its effect completely. Therefore, in the case of cluster translation, it can be concluded that:

- Translation disambiguation has no effect on MLDC based on cluster translation.
- The number of selected terms may have weak effect on MLDC in the case of cluster translation.

From a practical viewpoint, this finding is important because these conclusions support the conjecture that more 'efficient' MLDC, in which fewer terms in cluster vectors are translated without any disambiguation operation, may also be 'relatively' effective.

However, a fundamental problem is the overall low effectiveness of MLDC based on cluster translation. Table 5.22 shows nMI scores in the case of not merging any cluster at the second stage. For example, when the threshold for clustering at the first stage was 0.06, monolingual DC by each language generated 29 clusters in English, 11 in French, 16 in German and 7 in Italian, and 63 clusters in total were obtained (see the far-right column of Table 5.17). If these clusters are not merged at the second stage, then the nMI score of a simple union of these monolingual clusters becomes 0.361, as indicated in Table 5.22. Note that each set of clusters by respective language is more homogeneous except for French (with nMI scores of 0.474, 0.355, 0.482 and 0.461, respectively, as shown in Table 5.17), but the nMI score of the union of these sets reduces to 0.361 due to simply concatenating them into a single set. Unfortunately, all nMI scores of MLDC based on cluster translation indicated in Tables 5.18 and 5.20 are not higher than those of the simple

Table 5.20: Results of MLDC by cluster translation with disambiguation based on term co-occurrence statistics

	Thresholds at 1st stage (cluster generation)							
Thresholds at	0.06		0.08		0.10		0.12	
2nd stage	L	nMI	L	nMI	L	nMI	L	nMI
(a) No. of selected terms = 10								
0.06	48	0.339	124	0.339	221	0.316	312	0.299
0.08	49	0.341	132	0.348	252	0.323	389	0.313
0.10	51	0.344	146	0.355	294	0.336	446	0.323
0.12	57	0.351	155	0.358	332	0.344	535	0.331
Ave.	51.3	0.343	139.3	0.350	274.8	0.330	420.5	0.317
(b) No. of selected terms = 30								
0.06	48	0.333	125	0.346	230	0.321	303	0.301
0.08	53	0.344	149	0.357	306	0.339	448	0.324
0.10	57	0.347	158	0.359	359	0.347	564	0.332
0.12	62	0.357	167	0.363	391	0.353	654	0.339
Ave.	55.0	0.345	149.8	0.356	321.5	0.340	492.3	0.324
(c) No. of selected terms = 50								
0.06	51	0.343	130	0.349	220	0.319	262	0.296
0.08	54	0.349	147	0.352	304	0.339	421	0.317
0.10	60	0.356	158	0.359	354	0.344	562	0.334
0.12	61	0.357	170	0.367	385	0.351	672	0.345
Ave.	56.5	0.351	151.3	0.357	315.8	0.338	479.3	0.323
(d) No. of selected terms = 100								
0.06	45	0.327	109	0.341	157	0.298	152	0.261
0.08	54	0.344	143	0.362	254	0.325	280	0.293
0.10	58	0.353	158	0.366	326	0.337	464	0.324
0.12	62	0.358	168	0.367	378	0.346	614	0.340
Ave.	54.8	0.345	144.5	0.359	278.8	0.327	377.5	0.305

Note: Word selection was executed only at second stage.

Table 5.21: Results of ANOVA for data of MLDC based on cluster translation

	Variation	d.f.	Variance	Observed var.	P-value
Disambiguation	0.0004980	1	0.000498	0.906391	0.343
No. of selected terms	0.0027842	3	0.000928	1.689157	0.173
Interaction	0.0009396	3	0.000313	0.570059	0.636
Residual	0.0659314	120	0.000549		
Total	0.0701532	127			

Table 5.22: Score of nMI in the case of not merging any cluster at the second stage

	Thresholds at 1st stage			
	0.06	0.08	0.10	0.12
Score of nMI	0.361	0.375	0.367	0.363
No. of clusters	63	185	464	867

Note: Nos. of clusters are shown in Table 5.17.

unions. This means that there was no effect of MLDC based on cluster translation in comparison with monolingual DC.

Figure 5.8 shows a change of nMI score with different values of threshold at the second stage (only when 50 terms were selected and the threshold at the first stage was 0.08). The score of nMI increases up to around 0.1 of threshold at the second stage, but the score is in saturation at the area of higher threshold in which any translated cluster is not merged with the other (the nMI score is indicated in Table 5.22, i.e., 0.375). Upon checking the effectiveness further, very few cases where MLDC slightly outperformed the simple union were found (for example, in the case of cluster translation with disambiguation with a threshold of 0.12 in the first stage and a threshold of 0.35 in the second stage, its nMI score was 0.3628 in comparison with 0.3627 in the simple union), but their differences were very small. Therefore, it can be concluded that:

- MLDC based on cluster translation has almost no effect in comparison with monolingual DC.

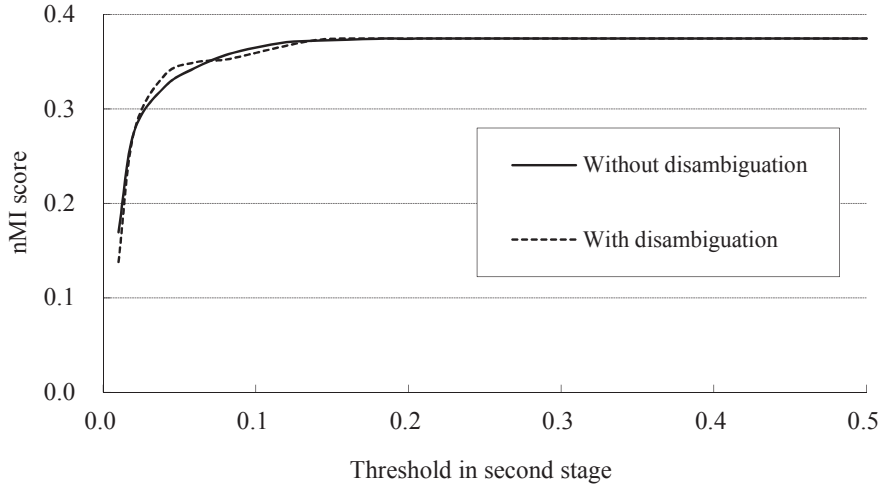


Figure 5.8: Change of nMI score with different values of threshold in second stage (no. of selected terms is 50 and threshold at first stage is 0.08)

Document translation

Tables 5.23 and 5.24 show the nMI scores of MLDC based on document translation without disambiguation and with disambiguation, respectively. In the case of document translation, the set of threshold θ_s was determined as $\{0.04, 0.06, 0.08, 0.10, 0.12, 0.14, 0.20, 0.25, 0.30, 0.35, 0.40\}$ so the peak of nMI scores can be roughly grasped, while the numbers of selected terms are identical with those in cluster translation (i.e., $\{10, 30, 50, 100\}$). In the case of higher values of threshold, many more clusters were generated than cluster translation. For example, when the threshold was 0.40, over 7,000 clusters were obtained (see Tables 5.23 and 5.24). Because the number of 'true' clusters is only 69, it may be able to consider that the higher threshold values did not provide successful results, but these cases will be included in the successive analysis for understanding an overall trend.

Table 5.23: Results of MLDC by document translation without disambiguation

Thresholds for clustering	No. of selected term 10		30		50		100	
	L	nMI	L	nMI	L	nMI	L	nMI
0.04	335	0.280	145	0.359	66	0.401	18	0.345
0.06	591	0.300	362	0.355	181	0.393	69	0.413
0.08	903	0.313	675	0.360	405	0.383	179	0.414
0.10	1258	0.321	1053	0.359	743	0.375	399	0.405
0.12	1643	0.326	1455	0.358	1166	0.377	675	0.397
0.14	2063	0.329	1914	0.359	1614	0.376	1065	0.391
0.20	3356	0.338	3466	0.356	3178	0.368	2557	0.379
0.25	4395	0.340	4715	0.353	4508	0.361	3968	0.369
0.30	5438	0.340	5863	0.348	5741	0.354	5292	0.360
0.35	6420	0.339	6986	0.344	6871	0.347	6504	0.351
0.40	7235	0.337	7892	0.339	7877	0.341	7668	0.343
Ave.	-	0.324	-	0.354	-	0.371	-	0.379

Table 5.24: Results of MLDC by document translation with disambiguation

Thresholds for clustering	No. of selected term 10		30		50		100	
	L	nMI	L	nMI	L	nMI	L	nMI
0.04	343	0.295	197	0.372	82	0.407	24	0.390
0.06	596	0.309	428	0.373	245	0.403	86	0.412
0.08	877	0.322	781	0.371	534	0.392	228	0.417
0.10	1214	0.327	1197	0.372	970	0.385	508	0.404
0.12	1592	0.334	1669	0.370	1424	0.384	900	0.400
0.14	2038	0.337	2202	0.369	1920	0.382	1381	0.394
0.20	3536	0.344	3779	0.363	3501	0.371	2982	0.381
0.25	4740	0.344	5032	0.356	4760	0.362	4284	0.370
0.30	5904	0.343	6137	0.350	5953	0.354	5539	0.360
0.35	6843	0.340	7162	0.344	7013	0.347	6670	0.351
0.40	7650	0.338	8025	0.339	7928	0.341	7775	0.343
Ave.	-	0.330	-	0.362	-	0.375	-	0.384

Table 5.25: Results of ANOVA for data of MLDC based on document translation

	Variation	d.f.	Variance	Observed var.	P-value
Disambiguation	0.0008097	1	0.0008097	2.1601399	0.146
No. of selected terms	0.0377586	3	0.0125862	33.5762425	0.000
Interaction	0.0000433	3	0.0000144	0.0384827	0.990
Residual	0.0299884	80	0.0003749		
Total	0.0686000	87			

First, unlike cluster translation, MLDC clearly has a positive effect in comparison with the simple union of clusters from monolingual DC. For example, the average of nMI scores over thresholds in the case of 100 terms selected is 0.379 without disambiguation (see Table 5.23) and is 0.384 with disambiguation (see Table 5.24), both of which are larger than 0.363 of the simple union (Table 5.22). Therefore, it can be concluded that:

- MLDC based on document translation has a substantial positive effect.

Also, unlike cluster translation, both the number of selected terms and translation disambiguation appear to influence the effectiveness of MLDC in the case of document translation. For example, the averages of nMI scores over thresholds were 0.324, 0.354, 0.371 and 0.379 in the case without disambiguation, and 0.330, 0.362, 0.375 and 0.384 with disambiguation (see the bottom lines of Tables 5.23 and 5.24). These four values in each sequence correspond to the numbers of selected terms, 10, 30, 50 and 100, respectively, and it is clear from a comparison between the two sequences that MLDC clustering with disambiguation was more effective than without disambiguation and that selecting many more words outperforms fewer words.

In order to obtain more evidence, ANOVA with two factors, disambiguation and the number of selected terms, was executed again using the scores in Tables 5.23 and 5.24 similarly to ANOVA in the case of cluster translation. Table 5.25 indicates the results. The p-values suggest that the number of selected terms has a strong effect (i.e., $p = 0.000$), which means that longer vectors of documents are desirable in MLDC based on document translation. In contrast, disambiguation does not have a statistically significant influence on the effectiveness of MLDC. However, its p-value is relatively small ($p = 0.146$), and therefore it would be difficult to reject completely that disambiguation has a positive effect on the performance of MLDC based on document translation. From the ANOVA results, in the case of document translation, it can be concluded that:

- Translation disambiguation has a weak positive effect on MLDC by document translation.
- Selecting many more terms for document translation enhances the effectiveness of MLDC.

As shown in Table 5.23 and 5.24, in the cases that 50 and 100 terms were selected, the number of generated clusters is close to that of 'true' clusters at thresholds providing higher scores of nMI. For example, in Table 5.23, when 100 terms were selected, the number is 69 at a threshold of 0.06 (nMI is 0.413), and in Table 5.24, when 50 terms, it is 82 at 0.04 (nMI is 0.407). This means that document translation strategy provides good clustering results. However, it is concurrently suggested that an appropriate threshold has to be specified according to the number of selected terms for obtaining a partition in which the number of clusters is reasonable.

5.4.6 Discussion

The results in this experiment are summarized in Table 5.26. Note that the results were obtained by using only a single dataset and particular bilingual dictionaries downloaded from the Internet. Therefore, if other datasets and more sophisticated language resources are used, then some of the conclusions may change [19]. For example, the conclusion that MLDC based on cluster translation has almost no effect in comparison with the simple union of clusters generated by monolingual DC may be rejected when using a more sophisticated language resource in another experiment, namely, cluster translation may be more effective.

[19] Of course, this is relevant to all experiments of IR and DC more or less. Rather, it would be important to accumulate such experimental results for enhancing the theory and technology of them.

Table 5.26: Summary of results from experiment

	Cluster translation	Document translation
Effect in comparison with monolingual clustering	No effect	Positive effect
Effect of translation disambiguation	No effect	Weak positive effect
Effect of many more terms to be selected for constructing vectors	Weak effect	Positive effect

However, the conclusion that document translation outperforms cluster translation would remain unchanged because a more sophisticated resource would improve document translation concurrently. For a similar reason, the conclusion that selecting many more terms has a positive effect on MLDC by document translation would be relatively reliable. Although a more sophisticated dictionary may be able to achieve better effectiveness by using fewer words, there is no good reason why longer document vectors would greatly reduce its effectiveness in the situation.

Of course, 'feature selection' is another research issue. For example, a comparison of the effectiveness between using all terms in a document and only a portion of them as the set of vector elements is beyond the scope of this thesis. Rather, this experiment focused on the number of terms entered into the translation process because translation disambiguation takes a long time when many source terms have to be processed. Even without disambiguation, it is desirable for efficient processing to translate fewer source terms, because dictionary-based translation often yields many translations (see Table 5.19). Therefore, the conclusion that longer vectors show higher effectiveness is not useful for developing actual systems, unfortunately.

In contrast, the fact that translation disambiguation had no large effect in this experiment (see Table 5.26) may make it easier to implement MLDC systems. In CLIR, disambiguation is often applied to translations of a given query that is much shorter than a document, in which ambiguous words play an important role in matching operations for IR. However, in the case of document translation or cluster translation, the relative importance of ambiguous terms would decrease because many other terms are included in the vectors. Non-ambiguous terms or combinations of ambiguous terms in a document or a cluster are expected to be useful for computing the similarity between each document and clusters.

Cluster translation would be effective only when several constraints are satisfied: 1) the translation quality is quite good, 2) subject representations of clusters are sufficiently appropriate, 3) the clustering algorithm is effective (in the case of LFC algorithm, an optimal value of threshold θ_s has to be correctly selected). The low performance of DC based on cluster translation in this experiment may be partially due to the simple representation or weighting of cluster vector. As mentioned above, in the case of cluster translation strategy, it is expected that the processing time of cluster translation is shorter than that of document translation if the number of clusters generated at the first stage is enough small. Although a clustering algorithm has to be executed two times in the cluster translation strategy, reduction in time for translation is attractive for actual MLDC. If more sophisticated representation or weighting of cluster vector allowing for effective clustering is developed and an effective method for reducing computational complexity in execution of clustering at the second stage is applied (e.g., by feature selection), then cluster translation may become an efficient strategy for MLDC.

In order to execute the LFC, threshold θ_s has to be determined a priori, which may cause a problem, as mentioned above. For example, in Tables 5.18, 5.20, 5.23 and 5.24, inappropriate values of the threshold produce low scores of nMI. This means that actual users have to know appropriate θ_s for obtaining good results of clustering in real applications, but it is usually difficult. If the number of clusters L is known a priori, then it is possible to use a k-means algorithm. However, L is also unknown in many DC situations. Which of θ_s and L is easy to predefine depends on each situation. This problem is a classical issue of DC and further researches are needed.

Chapter 6

Conclusion

In this thesis, first, CLIR methods (Chapter 2) and DC techniques (Chapters 3 and 4) have been exhaustively reviewed to identify useful techniques for large-scale MLDC in terms of efficiency. Particularly, regarding DC techniques, in addition to reviewing theories and algorithms systematically, small experiments were executed for obtaining deeper insights on the DC techniques, which are useful for filtering out DC techniques. Specifically, they were divided into two groups, and in Chapter 3, traditional techniques of hierarchical and non-hierarchical clustering were mainly focused on. Meanwhile, probabilistic and matrix-based clustering methods, which have been developed recently, were examined carefully in Chapter 4. Main DC techniques discussed in Chapters 3 and 4 are as follows: hierarchical clustering (single linkage, complete linkage, group average linkage and Ward's method), k-means clustering (basic algorithm, Hartigan-Wong method, online k-means, scalable k-means and spherical k-means), SKWIC method, leader-follower clustering, Scatter/Gather, self-organizing map (SOM), fuzzy clustering, probabilistic mixture model (Poisson, multinomial and von Mises-Fisher distributions), Bayesian model of multinomial mixture, probabilistic latent semantic analysis (PLSA), latent Dirichlet allocation (LDA), hierarchical Dirichlet process (HDP) mixture model, latent semantic indexing (LSI), principal component analysis (PCA), nonnegative matrix factorization (NMF), spectral clustering.

All translation methods in CLIR and all clustering techniques for textual data discussed in literature are not always appropriate to large-scale document collection. This thesis selected two components of the MLDC system through discussions in Chapter 2, 3 and 4, namely,

- dictionary-based translation with term disambiguation based on term co-occurrence frequency,
- double-pass leader-follower clustering algorithm,

in terms of efficiency and feasibility in clustering of large-scale document collections. A combination of the translating and clustering techniques was considered as the most promising scheme that enables large-scale MLDC in this thesis.

Chapter 5 reported three experiments examining effectiveness and efficiency of the MLDC system empirically as follows:

- Experiment of term disambiguation techniques. It was observed that 'best cohesion' method works well although its computational complexity is relatively low.
- Experiment of monolingual document clustering. It was shown that the leader-follower clustering algorithm for which the target file is scanned only twice can generate 'good' cluster sets, which are comparable with those obtained by the spherical k-means algorithm and the HDP mixture model.
- Experiment of multilingual document clustering. It was clarified that the MLDC system works well for a document collection including over 13,000 news articles written in English, French, German and Italian.

The spherical k-means algorithm and the HDP mixture model used in the second experiment require iterative computation (i.e., higher computational complexity), but they were selected as methods providing 'baseline' for verifying effectiveness of more 'efficient' leader-follower clustering algorithm. Also, in the second experiment, it was empirically shown that the leader-follower clustering algorithm works well even if target document sets are enlarged.

The last experiment investigating effectiveness of MLDC using dictionary-based translation and the double-pass clustering algorithm is the main experiment of this thesis, and the following conclusions were obtained:

- Document translation has a positive effect on MLDC, but cluster translation has almost no effect.
- A longer vector for matching operation in MLDC is desirable in the case of the document translation strategy.
- Translation disambiguation does not have a large effect on MLDC.

These findings would be valuable for developing more sophisticated MLDC systems.

In order to increment more our knowledge on MLDC and to enhance the algorithm for it, the following efforts would be indispensable:

- Experiment of MLDC using other document collections and language resources.
- Development of document and cluster representations more appropriate to MLDC.
- Application of more sophisticated CLIR techniques (e.g., multilingual IR in Section 2.2.6).
- Sophistication of the leader-follower clustering method based on techniques of text stream clustering (Section 3.3.7).

These tasks remains for future research efforts.

Bibliography

[1] M. Adriani. Using statistical term similarity for sense disambiguation in cross-language information retrieval. *Information Retrieval*, 2(1):71–82, 2000.

[2] M. Adriani. English-Dutch CLIR using query translation techniques. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 219–225. Springer-Verlag, 2002. (LNCS 2406).

[3] C. C. Aggarwal and P. S. Yu. On clustering massive text and categorical data streams. *Knowledge and Information Systems*, 24:171–196, 2010.

[4] C. C. Aggarwal and C.-X. Zhai. A survey of text clustering algorithm. In C. C. Aggarwal and C.-X. Zhai, editors, *Mining Text Data*, pages 77–128. Springer, 2012.

[5] J. Allan. Introduction to Topic Detection and Tracking. In J. Allan, editor, *Topic Detection and Tracking*, pages 1–16. Kluwer, 2002.

[6] J. Allan, R. Papka, and V. Lavrenko. On-line new event detection and tracking. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 37–45, 1998.

[7] L. AlSmait and C. Domeniconi. Text clustering with local semantic kernels. In M. W. Berry and M. Castellanos, editors, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, pages 88–105. Springer, 2008.

[8] G. Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

[9] M. R. Anderberg. *Cluster Analysis for Applications*. Academic press, 1973.

[10] R. K. Ando. Latent semantic space: iterative scaling improves precision of inter-document similarity measurement. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 216–223, 2000.

[11] R. K. Ando and L. Lee. Iterative residual rescaling: an analysis and generalization of LSI. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–162, 2001.

[12] N. O. Andrews and E. A. Fox. Recent developments in document clustering. Technical report, TR-07-35, Computer Science, Virginia Tech, 2007.

[13] R. Angelova and S. Siersdorfer. A neighborhood-based approach for clustering of linked document collections. In *CIKM' 06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 778–779, 2006.

[14] B. W. Bader, M. W. Berry, and A. N. Langville. Nonngeative matrix and tensor factorization for discussion tracking. In A. N. Srivastava and M. Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, pages 95–120. CRC Press, 2009.

[15] L. Ballesteros and W. B. Croft. Phrasal translation and query expansion techniques for cross-language information retrieval. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 84–91, 1997.

[16] L. Ballesteros and W. B. Croft. Resolving ambiguity for cross-language retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 64–71, 1998.

[17] L. A. Ballesteros. Cross-language retrieval via transitive translation. In W. B. Croft et al., editor, *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, pages 203–234. Kluwer, 2000.

[18] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra. Text clustering with mixture of von Mises-Fisher distribution. In A. N. Srivastava and M. Sahami, editors, *Text Mining: Classification, Clustering, and Applications*, pages 121–153. Chapman & Hall, 2009.

[19] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

[20] A. Banerjee and J. Ghosh. Frequency-sensitive competitive learning for scalable balanced clustering on high-dimensional hyperspheres. *IEEE Transactions on Neural Networks*, 15(3):702–719, 2004.

[21] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 19–26, 2002.

[22] P. Berkhin. Survey of clustering data mining techniques. Technical report, 2002.

[23] M. W. Berry and P. G. Young. Using latent semantic indexing for multilanguage information retrieval. *Computers and the Humanities*, 29(6):413–429, 1995.

[24] N. Bertoldi and M. Federico. Statistical models for monolingual and bilingual information retrieval. *Information Retrieval*, 7(1-2):53–72, 2004.

[25] J. C. Bezdek, C. Coray, R. Gunderson, and J. Watson. Detection and characterization of cluster substructure. *SIAM Journal on Applied Mathematics*, 40(2):339 – 372, 1981.

[26] G. W. Bian and H. H. Chen. Integrating query translation and document translation in a cross-language information retrieval system. In D. Farwell et al., editor, *Machine Translation and the Information Soup*, pages 250–265. Springer, 1998. (LNCS 1529).

[27] G. W. Bian and C. C. Lin. Trans-EZ at NTCIR-2 : synset co-occurrence method for English-Chinese cross-lingual information retrieval. In *Proceedings of the Second NTCIR Workshop on Research in Chinese & Japanese Text Retrieval and Text Summarization*, 2001. Available from: http://research.nii.ac.jp/ntcir/workshop/.

[28] D. M. Blei, T. L. Griffiths, and M. I. Jordan. The nested Chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2), 2010.

[29] D. M. Blei and J. D. Lafferty. Topic models. In A. N. Srivastava and M. Sahami, editors, *Text Mining: Classification, Clustering, and Application*, pages 71–93. CRC Press, 2009.

[30] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[31] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *Artifical Intelligence Review*, 13(5,6):365–391, 1999.

[32] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partition-based clustering for Web document categorization. *Decision Support Systems*, 27(3):329–341, 1999.

[33] A. Bookstein and W. Cooper. A general mathematical model for information retrieval system. *Library Quarterly*, 46(2):153–167, 1976.

[34] A. Bookstein and D. R. Swanson. Probabilistic models for automatic indexing. *American Documentation*, 16(4):283–289, 1965.

[35] V. P. G. Bote, F. de Moya Anegón, and V. H. Solana. Document organization using Kohonen's algorithm. *Informatoin Processing and Management*, 38(1):79–89, 2002.

[36] M. Boughanem, C. Chrisment, and N. Nassr. Investigation on disambiguation in CLIR: aligned corpus and bi-directional translation-based strategies. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 158–168. Springer-Verlag, 2002. (LNCS 2406).

[37] P. S. Bradley, U. M. Fayyad, and C. A. Reina. Scailing clustering algorithms to large database. In *Proceedings of the 4th International Conference on Knowledge Discovery & Data Mining (KDD98)*, pages 9–15, 1998.

[38] M. Braschler. Combination approaches for multilingual text retrieval. *Information Retrieval*, 7(1-2):183–204, 2004.

[39] M. Braschler and C. Peters. Cross-Language Evaluation Forum: objectives, results, achievements. *Information Retrieval*, 7(1-2):7–31, 2004.

[40] M. Braschler and P. Schäuble. Using corpus-based approaches in a system for multilingual information retrieval. *Information Retrieval*, 3(3):273–284, 2000.

[41] M. Braschler and P. Schäuble. Experiments with the Eurospider retrieval system for CLEF 2000. In C. Peters, editor, *Cross-Language Information Retrieval Evaluation*, pages 140–148. Springer-Verlag, 2001. (LNCS 2069).

[42] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, and R. L. Mercer. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[43] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC 3. In *Overview of the Third Text Retrieval Conference (TREC3)*. National Institute of Standards and Technology, 1995.

[44] C. Buckley and E. M. Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 25–32, 2004.

[45] C. Buckley, J. Walz, M. Mitra, and C. Cardie. Using clustering and superconcepts within SMART: TREC-6. In *Proceedings of TREC-6*. National Institute of Standards and Technology, 1998. Available from: http://trec.nist.gov/pubs.html.

[46] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 33–40, 2000.

[47] J. P. Callan, W. B. Croft, and J. Broglio. TREC and TIPSTER experiments with INQUERY. *Information Processing & Management*, 31(3):327–343, 1995.

[48] J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–28, 1995.

[49] F. Can. Incremental clustering for dynamic information processing. *ACM Transactions on Information Systems*, 10(2):143–164, 1993.

[50] F. Can, E. A. Fox, C. D. Snavely, and R. K. France. Incremental clustering for very large document databases: initial marian experience. *Information Sciences*, 84:101–114, 1995.

[51] F. Can and E. A. Ozkarahan. Two partitioning type clustering algorithms. *Journal of the American Society for Information Science*, 35(5):268–276, 1984.

[52] F. Can and E. A. Ozkarahan. Similarity and stability analysis of the two partitioning type clustering algorithms. *Journal of the American Society for Information Science*, 36(1):3–14, 1985.

[53] F. Can and E. A. Ozkarahan. Computation of term/document discrimination values by use of the cover coefficient concept. *Journal of the American Society for Information Science*, 38(3):171–183, 1987.

[54] F. Can and E. A. Ozkarahan. Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems*, 15(4):483–517, 1990.

[55] Y. Cao and H. Li. Base noun phrase translation using web data and the EM algorithm. In *Proceedings of the 19th International Conference on Computational Linguistics*, volume 1, pages 1–7, 2002.

[56] R. Chau, C. Yeh, and K. Smith. A neural network model for hierarchical multilingual text categorization. In J. Wang, X. Liao, and Z. Yi, editors, *Advances in Neural Networks - ISNN 2005*, pages 238–245. Springer-Verlag, 2005.

[57] A. Chen and F. C. Gey. Experiments on cross-language and patent retrieval at NTCIR-3 workshop. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, 2003. Available from: http://research.nii.ac.jp/ntcir/.

[58] A. Chen and F. C. Gey. Multilingual information retrieval using machine translation, relevance feedback and decompounding. *Information Retrieval*, 7(1-2):149–182, 2004.

[59] H. Chen, P. Hsu, R. Orwig, L. Hoopes, and F. Nunamaker Jr. Automatic concept classification of text from electronic meetings. *Communications of the ACM*, 37(10):56–73, 1994.

[60] H. H. Chen and C. J. Lin. A multilingual news summarizer. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 1*, pages 159–165, 2000.

[61] H.H. Chen and L. W. Ku. An NLP & IR approach to topic detection. In J. Allan, editor, *Topic Detection and Tracking: Event-based Information Organization*, pages 243–264. Kluwer, 2002.

[62] P.-J. Cheng, J.-W. Teng, R.-C. Chen, J.-H. Wang, W.-H. Lu, and L.-F. Chien. Translating unknown queries with Web corpora for cross-language information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 146–153, 2004.

[63] E. K. P. Chong and S. H. Żak. *An Introduction to Optimization*. Wiley, third edition, 2008.

[64] K. W. Church and P. Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990.

[65] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari. *Nonnegative Martrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. John Wiley and Sons, 2009.

[66] C. Cieri, S. Strassel, D. Graff, N. Martey, K. Rennert, and M. Liberman. Corpora for Topic Detection and Tracking. In J. Allan, editor, *Topic Detection and Tracking*, pages 33–66. Kluwer, 2002.

[67] C. W. Cleverdon. On the inverse relationships of recall and precision. *Journal of Documentation*, 28(2):93–107, 1972.

[68] C. W. Cleverdon and J. Mills. The testing of index language devices. *Aslib Proceddings*, 15(4):106–130, 1963.

[69] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[70] W. B. Croft. Clustering large files of documents using the single-link method. *Journal of the American Society for Information Science*, 28(6):341–344, 1977.

[71] D. B. Crouch. A file organization and maintenance procedure for dynamic document collections. *Information Processing & Management*, 11(1/2):11–21, 1975.

[72] D. R. Cutting, D. R. Karger, J. O. Pedersen, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[73] F. J. Damerau. An experiment in automatic indexing. *American Documentation*, 16(4):283–289, 1965.

[74] K. Darwish and D. W. Oard. Probabilistic structured query methods. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 338–344, 2003.

[75] M. Davis. New experiments in cross-language text retrieval at NMSU's Computing Research Lab. In *Proceedings of TREC-5*. National Institute of Standards and Technology, 1997. Available from: http://trec.nist.gov/pubs.html.

[76] M. Davis and T. Dunning. A TREC evaluation of query translation methods for multi-lingual text retrieval. In *Proceedings of TREC-4*. National Institute of Standards and Technology, 1995. Available from: http://trec.nist.gov/pubs.html.

[77] M. W. Davis. On the effective use of large parallel corpora in cross-language text retrieval. In G. Grefenstette, editor, *Cross-Language Information Retrieval*, pages 12–22. Kluwer, 1998.

[78] M. W. Davis and W. C. Ogden. QUILT: implementing a large-scale cross-language text retrieval system. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 92–98, 1997.

[79] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[80] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 269–274, 2001.

[81] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42:143–175, 2001.

[82] A. Diekema, F. Oroumchian, P. Sheridan, and E. D. Liddy. TREC7 evaluation of conceptual interlingua document retrieval (CINDOR) in English and French. In *Proceedings of TREC-7*. National Institute of Standards and Technology, 1999. Available from: http://trec.nist.gov/pubs.html.

[83] R. Diestel. *Graph Theory*. Springer, 2010.

[84] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, second edition*. Wiley, 2001.

[85] S. T. Dumais, T. K. Landauer, and M. L. Littman. Automatic cross-linguistic information retrieval using latent semantic indexing. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 16–23, 1996.

[86] J. C. Dunn. A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. *Journal of Cybernetics*, 3(3):32–57, 1973.

[87] T. Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74, 1993.

[88] D. Eichmann, M. E. Ruiz, and P. Srinivasan. Cross-language information retrieval with the UMLS metathesaurus. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 72–80, 1998.

[89] D. K. Evans and J. L. Klavans. A platform for multilingual news summarization. Technical Reports CUCS-014-03, Department of Computer Science, Columbia University, 2003.

[90] F. Farnstrom, J. Lewis, and C. Elkan. Scalability for clustering algorithms revisited. *ACM SIGKDD Explorations Newletter*, 2(1):51–57, 2000.

[91] M. Federico and N. Bertoldi. Statistical cross-language information retrieval using N-best query translations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 167–174, 2002.

[92] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[93] M. Franz, J. Scott McCarley, and S. Roukos. Ad hoc and multilingual information retrieval at IBM. In *Proceedings of TREC-7*. National Institute of Standards and Technology, 1999. Available from: http://trec.nist.gov/pubs.html.

[94] M. Franz, J. Scott McCarley, and R. Todd Ward. Ad hoc, cross-language and spoken document information retrieval at IBM. In *Proceedings of TREC-8*. National Institute of Standards and Technology, 2000. Available from: http://trec.nist.gov/pubs.html.

[95] H. Frigui and O. Nasraoui. Simultaneous clustering and attribute discrimination. In *Proceedings of the IEEE Conference on Fuzzy Systems*, pages 158–163, 2000.

[96] H. Frigui and O. Nasraoui. Simultaneous clustering and dynamic keyword weighting for text documents. In M. W. Berry, editor, *Survey of Text Mining*, pages 45–72. Springer, 2004.

[97] A. Fujii and T. Ishikawa. Japanese/English cross-language information retrieval: exploration of query translation and transliteration. *Computers and the Humanities*, 35:389–420, 2001.

[98] C. Galvez and F. Moya-Anegón. Approximate personal name-matching through finite-state graphs. *Journal of the American Society for Information Science and Technology*, 58(13):1960–1976, 2007.

[99] D. Gamerman and H. F. Lopes. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. Chapman & Hall/CRC, 2006.

[100] J. Gao and J.-Y. Nie. A study of statistical models for query translation: finding a good unit of translation. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 194–201, 2006.

[101] J. Gao, J.-Y. Nie, J. Zhang, E. Xun, Y. Su, M. Zhou, and C. Huang. Improving query translation for cross-language information retrieval using statistical models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96–104, 2001.

[102] R. Gaujoux. *Using the package NMF*, 2010. R package version 0.5.02.

[103] R. Gaujoux and C. Seoighe. A flexible R package for nonnegative matrix factorization. *BMC Bioinformatics*, 11(1):367, 2010.

[104] F. C. Gey. Inferring probability of relevance using the method of logistic regression. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–231, 1994.

[105] F. C. Gey. Research to improve cross-language retrieval: position paper for CLEF. In C. Peters, editor, *Cross-Language Information Retrieval Evaluation*, pages 83–88. Springer-Verlag, 2001. (LNCS 2069).

[106] F. C. Gey, H. Jiang, A. Chen, and R. R. Larson. Manual queries and machine translation in cross-language retrieval and interactive retrieval with Cheshire II at TREC-7. In *Proceedings of TREC-7*. National Institute of Standards and Technology, 1999. Available from: http://trec.nist.gov/pubs.html.

[107] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of National Academic Science*, 99(12):7821–7826, 2002.

[108] T. Gollins and M. Sanderson. Improving cross language information retrieval with triangulated translation. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 90–95, 2001.

[109] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of National Academic Science of the United States of America*, 101(Suppl.1):5228–5235, 2004.

[110] C. D. Gull. Historical note: information science and technology: from coordinate indexing to the global brain. *Journal of the American Society for Information Science*, 38(5):338–366, 1987.

[111] L. K. Hansen, S. Sigurdsson, T. Kolenda, F. A. Nielsen, U. Kjems, and J. Larsen. Modeling text with generalizable Gaussian mixtures. In *In Proceedings of ICASSP'2000*, pages 3494–3497, 2000.

[112] A. F. Harding and P. Willett. Indexing exhaustivity and the computation of similarity matrices. *Journal of the American Society for Information Science*, 31(4):298–300, 1980.

[113] D. Harman. Ranking algorithms. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structure & Algorithms*, pages 363–392. PTR Prentice Hall, 1992.

[114] D. Harman. Beyond English. In E. M. Voorhees and D. K. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*, pages 153–181. MIT Press, 2005.

[115] D. K. Harman, editor. *Overview of the Third Text REtrieval Conference (TREC-3)*. National Institute of Standards and Technology, 1995.

[116] R. A. Harshman. Foundations of the PARAFAC procedure: models and conditions for an 'exanatory' multi-modal factor analysis. *UCLA Working Papers Phonet*, 16:1–84, 1970.

[117] S. P. Harter. A probabilistic approach to automatic keyword indexing; part I: on the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science*, 26(4):197–206, 1975.

[118] S. P. Harter. A probabilistic approach to automatic keyword indexing; part II: analgorithm for probabilistic indexing. *Journal of the American Society for Information Science*, 26(5):280–289, 1975.

[119] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979.

[120] D. A. Harville. *Matrix Algebra from a Statistician's Perspective*. Springer, 1997.

[121] Q. He, K. Chang, E.-P. Lim, and J. Zhang. Bursty feature representation for clustering text streams. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 491–496, 2007.

[122] H. S. Heaps. *Information Retrieval*. Academic Press, 1978.

[123] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 76–84, 1996.

[124] T. Hedlund, E. Airio, H. Keskustalo, R. Lehtokangas, A. Pirkola, and K. Järveline. Dictionary-based cross-language information retrieval: learning experiences from CLEF 2000-2002. *Information Retrieval*, 7(1-2):99–119, 2004.

[125] T. Hedlund, H. Keskustalo, A. Pirkola, E. Airio, and K. Järvelin. Utaclir@CLEF2001: effects of compound splitting and n-gram techniques. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 118–136. Springer-Verlag, 2002. (LNCS 2406).

[126] D. Hiemstra. A linguistically motivated probabilistic model of information retrieval. In C. Nikolaou and C. Stephanidis, editors, *Research and Advanced Technology for Digital Libraries*, pages 569–584. Springer, 1998. (LNCS 1513).

[127] D. Hiemstra. Multilingual domain modeling in Twenty-One: automatic creation of a bi-directional translation lexicon from a parallel corpus. In *Proceedings of the 8th CLIN Meeting*, pages 41–58, 1998.

[128] D. Hiemstra and W. Kraaij. Twenty-one at TREC-7: ad-hoc and cross-language track. In *Proceedings of TREC-7*. National Institute of Standards and Technology, 1999. Available from: http://trec.nist.gov/pubs.html.

[129] D. Hiemstra, W. Kraaij, R. Pohlmann, and T. Westerveld. Translation resources, merging strategies, and relevance feedback for cross-language information retrieval. In C. Peters, editor, *Cross-Language Information Retrieval Evaluation*, pages 102–115. Springer-Verlag, 2001. (LNCS 2069).

[130] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[131] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. Exploration of full-text databases with self-organizing maps. In *Proceedings of ICNN'96, IEEE International Conference on Neural Networks*, pages 56–61, 1996.

[132] A. Hoonlor, B. K. Szymanski, M. I. Zaki, and V. Chaoji. Document clustering with bursty information. *Computing and Informatics*, 31(6):1533–1555, 2012.

[133] D. A. Hull and G. Grefenstette. Querying across languages: a dictionary-based approach to multilingual information retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–57, 1996.

[134] N. Ide and J. Véronis. Introduction to the special issue on word sense disambiguation: the state of the art. *Computational Linguistics*, 24(1):1–40, 1998.

[135] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[136] M.-G. Jang, S. H. Myaeng, and S. Y. Park. Using mutual information to resolve query translation ambiguities and query term weighting. In *Proceedings of the 37th Annual Conference of the Association of Computational Linguistics*, pages 223–229, 1999.

[137] K. Järvelin and J. Kekäläinen. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–48, 2000.

[138] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaulation of IR techniques. *ACM Transactions on Information Systems*, 20:422–446, 2002.

[139] G. J. F. Jones and A. M. Lam-Adesina. Experiments with machine translation for bilingual retrieval. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 59–77. Springer-Verlag, 2002. (LNCS 2406).

[140] I.-S. Kang, S.-H. Na, and J.-H. Lee. Combination approaches in information retrieval: words vs. n-grams and query translation vs. document translation. In *Proceedings of the Fourth NTCIR Workshop on Research in Information Access Technologies Information Retrieval, Question Answering and Summarization*, 2005. Available from: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings4/.

[141] S. Kaski. Dimensionality reduction by random mapping: fast similarity computation for clustering. In *Proceedings of IJCNN'98, IEEE International Joint Conference on Neural Networks*, volume 1, pages 413–418, 1998.

[142] E. Michael Keen. Some aspects of proximity searching in text retrieval system. *Journal of Information Science*, 18(2):89–98, 1992.

[143] J. Kekäläinen and K. Järveline. The impact of query structure and query expansion on retrieval performance. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 130–137, 1998.

[144] A. Kent, M. M. Berry, F. U. Luehrs Jr., and J. W. Perry. Machine literature searching viii.: operational criteria for designing information retrieval systems. *American Documentation*, 6(2):93–101, 1955.

[145] K. Kishida. Mathematical formulations of Bradford's law derived from relatioships among bibliometric laws. *Library and Information Science*, (26):55–65, 1988. (in Japanese).

[146] K. Kishida. A single-link method algorithm for clustering large document collections. *Library and Information Science*, (47):27–38, 2002. (in Japanese).

[147] K. Kishida. Techniques of document clustering: a review. *Library and Information Science*, (49):33–75, 2003. (in Japanese).

[148] K. Kishida. Property of average precision and its generalization: an examination of evaluation indicator for information retrieval experiments. NII Technical Reports NII-2005-014E, NII, 2005.

[149] K. Kishida. Technical issues of cross-language information retrieval: a review. *Information Processing & Management*, 41(3):433–455, 2005.

[150] K. Kishida. Term disambiguation techniques based on target document collection for cross-language information retrieval: an empirical comparison of performance between techniques. *Information Processing & Management*, 43(1):103–120, 2007.

[151] K. Kishida. Prediction of performance of cross-language information retrieval using automatic evaluation of translation. *Library and Information Science Research*, 30(2):138–144, 2008.

[152] K. Kishida. High-speed rough clustering for very large document collections. *Journal of the American Society for Information Science and Technology*, 61(6):1092–1104, 2010.

[153] K. Kishida. Methods for cross-language information retrieval. In E. F. Caldwell, editor, *Bilinguals: Cognition, Education and Language Processing*, pages 243–286. Nova Science Publishers, 2010.

[154] K. Kishida. Double-pass clustering technique for multilingual document collections. *Journal of Information Science*, 37(3):304–321, 2011.

[155] K. Kishida, K.-H. Chen, S. Lee, K. Kuriyama, N. Kando, and H.-H. Chen. Overview of CLIR task at the Sixth NTCIR Workshop. In *Proceedings of the 6th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-Lingual Information Access*, 2007. Available from: http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings6/.

[156] K. Kishida and E. Ishita. Translation disambiguation for cross-language information retrieval using context-based translation probability. *Journal of Information Science*, 35(4):481–495, 2009.

[157] K. Kishida and N. Kando. Two-stage refinement of query translation in a pivot language approach to cross-lingual information retrieval: an experiment at CLEF 2003. In C. Peters et al., editor, *Comparative Evaluation of Multilingual Information Access Systems*, pages 253–262. Springer-Verlag, 2004. (LNCS 3237).

[158] K. Kishida and N. Kando. A hybrid approach to query and document translation using a pivot language for cross-language information retrieval. In C. Peters et al., editor, *Accessing Multilingual Information Repositories*, pages 93–101. Springer-Verlag, 2006. (LNCS 4022).

[159] K. Kishida, N. Kando, and K. H. Chen. Two-stage refinement of transitive query translation with english disambiguation for cross-language information retrieval: an experiment at CLEF2004. In C. Peters et al., editor, *Multilingual Information Access for Text, Speech and Images, Lecture Notes in Computer Science*, pages 135–142. Springer-Verlag, 2005. (LNCS 3491).

[160] J. Kleinberg. Bursty and hierarchical structure in streams. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 91–101, 2002.

[161] K. Knight and J. Graehl. Machine transliteration. *Computational Linguistics*, 24(4):599–612, 1998.

[162] M. Kobayashi and M. Aono. Vector space models for search and cluster mining. In M. W. Berry, editor, *Survey of Text Mining*, pages 103–122. Springer, 2004.

[163] P. Koehn and K. Knight. Estimating word translation probabilities from unrelated monolingual corpora using the EM algorithm. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 711–715, 2000.

[164] J. Kogan. *Introduction to Clustering Large and High-Dimensional Data*. Cambridge, 2007.

[165] T. Kohonen. *Self-Organizing Maps*. Springer-Verlag, 1995.

[166] T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11:574–585, 2000.

[167] W. Kraaij. TNO at CLEF-2001: comparing translation resources. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 78–93. Springer-Verlag, 2002. (LNCS 2406).

[168] W. Kraaij, R. Pohlmann, and D. Hiemstra. Twenty-one at TREC-8: using language technology for information retrieval. In *Proceedings of TREC-8*. National Institute of Standards and Technology, 2000. Available from: http://trec.nist.gov/pubs.html.

[169] K. Kummamuru, A. Dhawale, and R. Krishnapuram. Fuzzy co-clustering of documents and keywords. In *The 12th IEEE International Conference on Fuzzy Systems (FUZZ '03)*, volume 2, pages 772–777, 2003.

[170] J.-S. Kuo, H. Li, and Y.-K. Yang. Active learning for constructing transliteration lexicons from the Web. *Journal of the American Society for Information Science and Technology*, 59(1):126–135, 2008.

[171] K. Lagus and S. Kaski. Keyword selection method for characterizing text document maps. In *Proceedings of ICANN 99, the 9th International Conference on Neural Networks*, volume 1, pages 371–376, 1999.

[172] F. W. Lancaster. *Indexing and Abstracting in Theory and Practice*. Library Association, second edition, 1998.

[173] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical Review E*, 80, 2009. DOI: 10.1103/PhysRevE.80.056117.

[174] T. Landauer and M. L. Littman. A statistical method for language-independent representation of the topical content of text segments. In *Proceedings of the 6th Annual Conference of the UW Centre for the New Oxford English Dictionary and Text Research*, 1990. Available from: http://www.cs.duke.edu/~mlittman/docs/x-lang.ps.

[175] T. K. Landauer, D. S. McNamara, S. Dennis, and W. Kintsch, editors. *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[176] L. S. Larkey and M. E. Connell. Structured queries, language modeling, and relevance modeling in cross-language information retrieval. *Information Processing & Management*, 41(3):457–473, 2005.

[177] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 16–22, 1999.

[178] V. Lavrenko, M. Choquette, and W. B. Croft. Cross-lingual relevance models. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 175–182, 2002.

[179] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2001.

[180] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[181] K.-S. Lee, K. Kageura, and K.-S. Choi. Implicit ambiguity resolution using incremental clustering in cross-language information retrieval. *Information Processing & Management*, 40(1):145–159, 2004.

[182] T. Leek, R. Schwartz, and S. Sista. Probabilistic approaches to Topic Detection and Tracking. In J. Allan, editor, *Topic Detection and Tracking*, pages 67–83. Kluwer, 2002.

[183] L.J Leftin. Newsblaster Russian-English clustering performance analysis. Technical Reports CUCS-010-03, Department of Computer Science, Columbia University, 2003.

[184] R. Lehtokangas, E. Airio, and K. Järveline. Transitive dictionary translation challenges direct dictionary translation in CLIR. *Information Processing & Management*, 40(6):973–988, 2004.

[185] M. E. Lesk. Word-word associations in document retrieval systems. *American Documentation*, 20:27–28, 1969.

[186] G.-A. Levow, D. W. Oard, and P. Resnik. Dictionary-based techniques for cross-language information retrieval. *Information Processing & Management*, 41(3):523–547, 2005.

[187] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: a new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[188] W. C. Lin and H. H. Chen. Description of NTU approach to NTCIR3 multilingual information retrieval. In *Proceedings of the Third NTCIR Workshop on Research in Information Retrieval, Automatic Text Summarization and Question Answering*, 2003. Available from: http://research.nii.ac.jp/ntcir/.

[189] X. Lin. Map displays for information retrieval. *Journal of the American Society for Information Science*, 48(1):40–54, 1997.

[190] X. Lin, D. Soergel, and G. Marchionini. A self-organizing sematic map for information retrieval. In *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, 1991.

[191] M. L. Littman, S. Dumais, and T. K. Landauer. Automatic cross-language information retrieval using latent semantic indexing. In G. Grefenstette, editor, *Cross-Language Information Retrieval*, pages 51–62. Kluwer, 1998.

[192] S. Liu, F. Liu, C. Yu, and W. Meng. An effective approach to document retrieval via utilizing WordNet and recognizing phrases. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 266–272, 2004.

[193] X. Liu, Y. Gong, W. Xu, and S. Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 191–198, 2002.

[194] Y. Liu, R. Jin, and J. Y. Chai. A maximum coherence model for dictionary-based cross-language information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 536–543, 2005.

[195] Y.-B. Liu, J.-R. Cai, J. Yin, and A. W.-C. Fu. Clustering text data stream. *Journal of Computer Science and Technology*, 23(1):112–128, 2008.

[196] B. Long, Z. Zhang, and P. S. Yu. *Relational Data Clustering: Models, Algorithms, and Applications*. CRC Press, 2010.

[197] F. Lopez-Ostenero, J. Gonzalo, and F. Verdejo. Noun phrases as building blocks for cross-language search assistance. *Information Processing & Management*, 41(3):549–568, 2005.

[198] Y. Lu, Q. Mei, and C. Zhai. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14(2):178–203, 2011.

[199] H. P. Luhn. The automatic creation of literature abstracts. *IBM Journal Research and Development*, 2(2):159–165, 1958.

[200] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[201] A. Maeda, F. Sadat, M. Yoshikawa, and S. Uemura. Query term disambiguation for Web cross-language information retrieval using a search engine. In *Proceedings of the 5th International Workshop on Information Retrieval with Asian languages*, pages 25–32, 2000.

[202] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge, 2008.

[203] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 2000.

[204] K. Markey. Twenty-five years of end-user searching, part 1: resesearch findings. *Journal of the American Society for Information Science and Technology*, 58(8):1071–1081, 2007.

[205] K. Markó, S. Schulz, O. Medelyan, and U. Hahn. Bootstrapping dictionaries for cross-language information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 528–535, 2005.

[206] M. Markou and S. Singh. Novelty detection: a review - part 1: statistical approaches. *Signal Processing*, 83(12):2481–2497, 2003.

[207] B. Mathieu, R. Besançon, and C. Fluhr. Multilingual document clusters discovery. In *Proceedings of RIAO' 2004*, pages 1–10, 2004.

[208] G. McLachlan and T. Krishnan. *The EM Algorithm and Extensions*. John Wiley & Sons, second edition, 2008.

[209] G. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, 2000.

[210] P. McNamee and J. Mayfield. Comparing cross-language query expansion techniques by degrading translation resources. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 159–166, 2002.

[211] P. McNamee and J. Mayfield. JHU/APL experiments at CLEF: translation resources and score normalization. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 193–208. Springer-Verlag, 2002. (LNCS 2406).

[212] P. McNamee and J. Mayfield. Character n-gram tokenization for European language text retrieval. *Information Retrieval*, 7(1-2):73–97, 2004.

[213] I. Dan Melamed. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249, 2000.

[214] R. Mihalcea and D. Radev. *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press, 2011.

[215] D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden Markov model information retrieval system. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221, 1999.

[216] S. Montavalo, R. Martínez, A. Casillas, and V. Fresno. Multilingual news clustering: feature translation vs. identification of cognate named entities. *Pattern Recognition Letters*, 28:2305–2311, 2007.

[217] C. Monz and B. J. Dorr. Iterative translation disambiguation for cross-language information retrieval. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 520–527, 2005.

[218] I. Moulinier and H. Molina-Salgado. Thomson Legal and Regulatory experiments for CLEF2002. In C. Peters et al., editor, *Advances in Cross-Language Information Retrieval*, pages 155–163. Springer-Verlag, 2003. (LNCS 2785).

[219] R. M. Neal. Markov Chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[220] J.-Y. Nie. CLIR using a probabilistic translation model based on Web documents. In *Proceedings of TREC-8*. National Institute of Standards and Technology, 2000. Available from: http://trec.nist.gov/pubs.html.

[221] J.-Y. Nie and F. Jin. A multilingual approach to multilingual information retrieval. In C. Peters et al., editor, *Advances in Cross-Language Information Retrieval*, pages 101–110. Springer-Verlag, 2003. (LNCS 2785).

[222] J.-Y. Nie, M. Simard, P. Isabelle, and R. Durand. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the Web. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, 1999.

[223] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39:103–134, 2000.

[224] M. Nilsson. Hierarchical clustering using non-greedy principal direction divisive partitioning. *Information Retrieval*, 5:311–321, 2002.

[225] D. W. Oard and A. R. Diekema. Cross-language information retrieval. *Annual Review of Information Science and Technology*, 33:223–256, 1998.

[226] D. W. Oard and P. Hackett. Document translation for cross-language text retrieval at the University of Maryland. In *Proceedings of TREC-6*. National Institute of Standards and Technology, 1998. Available from: http://trec.nist.gov/pubs.html.

[227] D. W. Oard, G. A. Levow, and C. I. Cabezas. CLEF experiments at Maryland: statistical stemming and backoff translation. In C. Peters, editor, *Cross-Language Information Retrieval Evaluation*, pages 176–187. Springer-Verlag, 2001. (LNCS 2069).

[228] F. J. Och and H. Ney. Systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51, 2003.

[229] R. E. Orwig, H. Chen, and J. F. Nunamaker Jr. A graphical, self-organizing approach to classifying electronic meeting output. *Journal of the American Society for Information Science*, 48(2):157–170, 1997.

[230] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318. ACL, 2002.

[231] H. J. Peat and P. Willet. The limitations of term co-occuurence data for query expansion in document retrieval systems. *Journal of the American Society for Information Science*, 42(5):378–383, 1991.

[232] W. Pedrycz. *Knowledge-based Clustering*. John Wiley & Sons, 2005.

[233] J.-F. Pessiot, Y.-M. Kim, M. R. Amini, and P. Gallinari. Improving document clustering in a learned concept space. *Information Proceesing & Management*, 46:180–192, 2010.

[234] C. Peters and P. Sheridan. Multilingual information access. In M. Agosti, et al., editor, *Lectures on Information Retrieval*, pages 51–80. Springer-Verlag, 2001. (LNCS 1980).

[235] M.H. Pham, D. Bernhard, G. Diallo, R. Messai, and M. Simonet. SOM-based clustering of multilingual documents using an ontology. In H. O. Nigro, S. E. Gonzalez Cisaro, and D. H. Xodo, editors, *Data Mining with Ontologies: Implementations, Findings and Framework*, pages 65–82. Information Science Reference, 2008.

[236] A. Pirkola. The effects of query structure and dictionary-based cross-language information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–63, 1998.

[237] A. Pirkola, T. Hedlund, H. Keskustalo, and K. Järvelin. Dictionary-based cross-language information retrieval: problems, methods, and research findings. *Information Retrieval*, 4(3-4):209–230, 2001.

[238] A. Pirkola, J. Toivonen, H. Keskustalo, K. V., and K. Järvelin. Fuzzy translation of cross-lingual spelling variants. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–352, 2003.

[239] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

[240] M. F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[241] B. Pouliquen, R. Steinberger, C. Ignat, E. Kasper, and I. Temnikova. Multilingual and cross-lingual news topic tracking. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING)*, pages 959–965, 2004.

[242] A. L. Powell, J. C. French, J. Callan, M. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–239, 2000.

[243] Y. Qiu and H. Frei. Concept based query expansion. In *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 160–169, 1993.

[244] Y. Qu, G. Grefenstette, and D. A. Evans. Automatic transliteration for Japanese-to-English text retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 353–360, 2003.

[245] Y. Qu, G. Grefenstette, and D. A. Evans. Resolving translation ambiguity using monolingual corpora. In C. Peters et al., editor, *Advances in Cross-Language Information Retrieval*, pages 223–241. Springer, 2003. (LNCS 2785).

[246] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2011.

[247] E. Rasmussen. Clustering algorithms. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structure and Algorithms*, pages 419–442. PTR Prentice Hall, 1992.

[248] A. Rauber, M. Dittenbach, and D. Merkl. Toward automatic content-based organization of multilingual digital libraries: an english, french and german view of the russian information agency nowosti news. In *Proceedings of the Third All-Russian Scientific Conference - Digital Libraries: Advanced Methods and Technologies, Digital Collections (RCDL01)*, 2001.

[249] B. Rehder, M. L. Littman, S. Dumais, and T. K. Landauer. Automatic 3-language cross-language information retrieval with latent semantic indexing. In *Proceedings of the TREC6*. National Institute of Standards and Technology, 1998. Available from: http://trec.nist.gov/pubs.html.

[250] P. Resnik. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, 1999.

[251] L. Rigouste, O. Cappé, and F. Yvon. Inference and evaluation of the multinomoial mixture model for text clustering. *Information Processing & Management*, 43:1260–1280, 2007.

[252] C. P. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer, 2010.

[253] R. E. Robertson, C. J. van Rijsbergen, and M. F. Porter. Probabilistic models of indexing and seaching. In R. N. Oddy, et al., editor, *Information Retrieval Research*, pages 35–56. Butterworth, 1981.

[254] S. E. Robertson. On relevance weight estimation and query expansion. *Journal of Documentation*, 42(3):182–188, 1986.

[255] S. E. Robertson. On sample sizes for non-matched-pair IR experiments. *Information Processing & Management*, 26(6):739–753, 1990.

[256] S. E. Robertson, M. E. Maron, and W. S. Cooper. Probability of relevance: a unification of two competing models for document retrieval. *Information Technology: Research and Development*, 1:1–21, 1982.

[257] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[258] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

[259] S. E. Robertson, S. Walker, S. Jones, and M. M. Hancock-Beaulieu. Okapi at TREC-3. In *Overview of the Third Text REtrieval Conference (TREC3)*. National Institute of Standards and Technology, 1995. http://trec.nist.gov/pubs.html.

[260] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, 1971.

[261] M. Rogati and Y. Yang. Resource selection for domain-specific cross-lingual IR. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161, 2004.

[262] D. G. Roussinov and H. Chen. Information navigation on the web by clustering and summarizing query results. *Informatoin Processing and Management*, 37(0):789–816, 2001.

[263] M. J. Rufo, C. J. Pérez, and J. Martín. Bayesian analysis of finite mixtures of multinomial and negative-multinomial distributions. *Computational Statistics & Data Analysis*, 51:5452–5466, 2007.

[264] B. Saha and P. Mitra. Dynamic algorithm for graph clustering using minimum cut tree. In *ICDMW '06 Proceedings of the Sixth IEEE International Conference on Data Mining*, pages 667–671, 2006.

[265] G. Salton. Automatic processing of foreign language documents. *Journal of the American Society for Information Science*, 21(3):187–194, 1970.

[266] G. Salton. The SMART system 1961-1976: experiments in dynamic processing. In A. Kent, editor, *Encyclopedia of Library and Information Science, Vol.28*, pages 1–36. Marcel Dekker, 1980.

[267] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[268] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[269] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[270] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(1):613–620, 1975.

[271] T. Saracevic. Relevance: a review of and a framework for thinking on the notion in information science. *Journal of the American Society for Information Science*, 26(6):321–343, 1975.

[272] T. Saracevic. Relevance: a review of the literature and framework for thinking on the notion in information science, part II: nature and manifestations of relevance. *Journal of the American Society for Information Science and Technology*, 58(13):1915–1933, 2007.

[273] T. Saracevic. Relevance: a review of the literature and framework for thinking on the notion in information science, part III: behavior and effects of relevance. *Journal of the American Society for Information Science and Technology*, 58(13):2126–2144, 2007.

[274] J. Savoy. Report on CLEF-2001 experiments: effective combined query-translation approach. In C. Peters et al., editor, *Evaluation of Cross-Language Information Retrieval Systems*, pages 27–43. Springer-Verlag, 2002. (LNCS 2406).

[275] J. Savoy. Report on CLEF-2002 experiments: combining multiple sources of evidence. In C. Peters et al., editor, *Advances in Cross-Language Information Retrieval*, pages 66–90. Springer-Verlag, 2003. (LNCS 2785).

[276] J. Savoy. Combining multiple strategies for effective monolingual and cross-language retrieval. *Information Retrieval*, 7(1-2):121–148, 2004.

[277] L. Schamber. Relevance and information behavior. *Annual Review of Information Science and Technology*, 29:3–48, 1994.

[278] H. Schütze and C. Silverstein. Projections for efficient document clustering. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 74–81, 1997.

[279] J. Scott McCarley. Should we translate the documents or the queries in cross-language information retrieval? In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 208–214, 1999.

[280] R. Sedgewick. *Algorithms in C++: Parts1-4*. Addison Wesley, 3rd edition, 1998.

[281] H.-C. Seo, S.-B. Kim, H.-C. Rim, and S.-H. Myaeng. Improving query translation in English-Korean cross-language information retrieval. *Information Processing & Management*, 41(3):507–522, 2005.

[282] P. Sheridan and J. P. Ballerini. Experiments in multilingual information retrieval using the SPIDER system. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 58–64, 1996.

[283] J. Silva, J. Mexia, C. A. Coelho, and G. Lopes. Multilingual document clustering, topic extraction and data transformations. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence: Knowledge Extraction, Multi-agent Systems, Logic Programming, and Constraint Solving*, pages 74–87. Springer-Verlag, 2001.

[284] C. Silverstein and J. O. Pedersen. Almost-constant-time clustering of arbitrary corpus subsets. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 60–66, 1997.

[285] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, 1996.

[286] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–273, 2000.

[287] A. F. Smeaton, M. Burnett, F. Crimmins, and G. Quinn. An architecture for efficient document clustering and retrieval on a dynamic collection of newspaper texts. In *20th BCS-IRGS Colloquium on Information Retrieval*, 1998.

[288] K. Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.

[289] K. Sparck Jones. Retrieval system test 1958-1978. In K. Sparck Jones, editor, *Information Retrieval Experiment*, pages 213–255. Butterworth, 1981.

[290] K. Sparck Jones and C. J. van Rijsbergen. Information retrieval test collection. *Journal of Documentation*, 32(1):59–75, 1976.

[291] R. Sperer and D. W. Oard. Structured translation for cross-language information retrieval. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2000.

[292] P. Srinivasan. Thesaurus construction. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structure & Algorithms*, pages 161–218. PTR Prentice Hall, 1992.

[293] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[294] M. Steyvers and T. Griffiths. Probailistic topic model. In T. K. Landauer et al., editor, *Handbook of Latent Semantic Analysis*. Lawrence Erlbaum Associates, 2007.

[295] Z. Su, J. Kogan, and C. Nicholas. Constrained clustering with k-means type algorithm. In M. W. Berry and J. Kogan, editors, *Text Mining: Applications and Thoery*, pages 81–103. Wiley, 2010.

[296] Y. Sun, H. Deng, and J. Han. Probabilistic models for text mining. In C. C. Aggarwal and C.-X. Zhai, editors, *Mining Text Data*, pages 259–295. Springer, 2012.

[297] T. Talvensaari, J. Laurikkala, K. Järveline, and M. Juhola. A study on automatic creation of a comparable document collection in cross-language information retrieval. *Journal of Documentation*, 62(3):372–387, 2006.

[298] R. Tang and P. Solomon. Toward an understanding of the dynamics of relevance judgment: an analysis of one person's search behavior. *Information Processing & Management*, 34(2/3):237–256, 1998.

[299] H. Tanizaki. A simple Gamma random number generator for arbitrary shape parameters. *Economics Bulletin*, 3(7):1–10, 2008.

[300] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. Technical report, Department of Statisics, University of Berkeley, 2005.

[301] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[302] Y. W. Teh, K. Kurihara, and M. Welling. Collapsed variational inference for HDP. In *Advances in Neural Information Processing Systems*, volume 20, 2008.

[303] M. Thelwall. *Link Analysis: An Information Science Approach*. Elesevier, 2004.

[304] J. Toivonen, A. Pirkola, H. Keskustalo, K. Visala, and K. Järveline. Translating cross-lingual spelling variants using transformation rules. *Information Processing & Management*, 41(4):859–872, 2005.

[305] K. Tsuji and K. Kageura. Automatic generation of Japanese-English bilingual thesauri based on bilingual corpora. *Journal of the American Society for Information Science and Technology*, 57(7):891–906, 2006.

[306] H. R. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, 1991.

[307] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[308] E. M. Voorhees. The efficiency of inverted index and cluster searches. In *Proceedings of the 9th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 164–174, 1986.

[309] E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.

[310] E. M. Voorhees and D. K. Harman, editors. *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[311] J. Wang and D. W. Oard. Combining bidirectional translation and synonymy for cross-language information retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 202–209, 2006.

[312] S. Wartik. Boolean operations. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structure and Algorithms*, pages 264–292. PTR Prentice Hall, 1992.

[313] A. Webb. *Statistical Pattern Recognition*. Wiley, 2nd edition, 2002.

[314] C. P. Wei, C. C. Yang, and C. M. Lin. A latent semantic-based approach to multilingual document clustering. *Decision Support System*, 45:606–620, 2008.

[315] M. Welling and M. Weber. Positive tensor factrorization. *Pattern Recoginition Letters*, 22:1255–1261, 2001.

[316] John W. Wilbur. Non-parametric significance tests of retrieval performance comparison. *Journal of Information Science*, 20(4):270–284, 1994.

[317] P. Willett. Document clustering using an inverted file approach. *Journal of Information Science*, 2(5):223–231, 1980.

[318] P. Willett. A fast procedure for the calculation of similarity coefficients in automatic classification. *Information Processing & Management*, 17:53–60, 1981.

[319] K. Wu and B. L. Lu. Cross-lingual document clustering. In Z. H. Zhou, H. Li, and Q. Yang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 956–963. Springer-Verlag, 2007.

[320] J. Xu and R. Weischedel. Empirical studies on the impact of lexical resources on CLIR performance. *Information Processing & Management*, 41(3):475–487, 2005.

[321] J. Xu, R. Weischedel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 105–110, 2001.

[322] R. Xu and D. C. Wunsch II. *Clustering*. Wiley, 2009.

[323] W. Xu and Y. Gong. Document clustering by concept factorization. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 202–209, 2004.

[324] W. Xu, X. Liu, and Y. Gong. Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–273, 2003.

[325] K. Yamabana, K. Muraki, S. Doi, and S. Kamei. A language conversion front-end for cross-language information retrieval. In G. Grefenstette, editor, *Cross-Language Information Retrieval*, pages 93–104. Kluwer, 1998.

[326] J. P. Yamron, L. Gillick, and P. van Mulbregt. Statistical models of topical content. In J. Allan, editor, *Topic Detection and Tracking*, pages 115–134. Kluwer, 2002.

[327] Y. Yang, J. Carbonell, R. D. Brown, and R. E. Frederking. Translingual information retrieval: learning from bilingual corpora. *Artificial Intelligence*, 103:323–345, 1998.

[328] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–49, 1999.

[329] O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 45–54, 1998.

[330] O. Zamir and O. Etzioni. Grouper: a dynamic clustering interface to Web search results. In *The Eighth International World Wide Web Conference*, Tronto, Canada, 1999. http://www8.org/w8-papers/.

[331] D. Zeimpekis and E. Gallopoulos. PDDP(l): Towards a flexible principal direction divisive partitioning clustering algorithm. In *Proceedings of IEEE ICDM '03 Workshop on Clustering Large Data Sets*, pages 26–35, 2003.

[332] D. Zeimpekis and E. Gallopoulos. Principal direction divisive partitioning with kernels and k-means steering. In M. W. Berry and M. Castellanos, editors, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, pages 45–64. Springer, 2008.

[333] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, 2001.

[334] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1:141–182, 1997.

[335] X. Zhang, X. Hu, T. He, E. K. Park, and X. Zhou. Utilizing different link types to enhance document clustering based on Markov Random Field model with relaxation labeling. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 2012.

[336] Y. Zhang and P. Vines. Using the web for automated translation extraction in cross-language information retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 162–169, 2004.

[337] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document databases. In *Proceeding of the 2002 ACM CIKM*, pages 515–524, 2002.

[338] Y. Zhao and G. Karypis. Empirical and theoretical comparisons of selected criterion functions for document clustering. *Machine Learning*, 55:311–331, 2004.

[339] Y. Zhao and G. Karypis. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10:141–168, 2005.

[340] S. Zhong. Efficient streaming text clustering. *Neural Networks*, 18:790–798, 2005.

[341] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.

[342] S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8:374–384, 2005.

[343] G. K. Zipf. *Human Behavior and the Principle of Least Effort*. Addison-Wesley, 1949.