A Thesis for the Degree of Ph.D. in Engineering

Cost-Efficient Text Classification without Manually Labeled Data

September 2025

Graduate School of Science and Technology,
Keio University

Yejian Zhang

Abstract

Text classification helps manage large volumes of unstructured data. Its applications are manifold: analyzing software user feedback to guide developers in maintenance and evolution; performing sentiment analysis on customer reviews to help enterprises enhance products and services; detecting toxic comments to filter harmful content; and conducting topic categorization to organize and retrieve documents efficiently. However, most machine-learning-based approaches rely heavily on large amounts of manually labeled data for training. Moreover, when viewed through the lens of software engineering for machine learning, practical deployment faces additional hurdles, including the need for domain expertise and substantial computational resources. This dissertation investigates how to design cost-efficient and accessible ML-based text classification systems that require little or no manual labeling, while also aiming to lower the barriers related to computational resource consumption and the need for expertise.

Two complementary, cost-efficient text classification systems are presented in this dissertation. The first system trains a domain-adapted language model through self-supervised learning and combines unsupervised clustering with active learning to identify the most informative instances for labeling. This system enables accurate classification of software engineering-oriented game user reviews while requiring only a small number of labeled instances. The second system embeds a large language model as an oracle within an active learning loop, delivering cost-effective cross-task text classification—including sentiment analysis, news topic categorization, and toxic comment detection—without any manually labeled data.

The proposed systems were evaluated through comprehensive experiments. The first achieved 88.8% accuracy on Steam game reviews with minimal supervision and demonstrated superior cost-effectiveness. The second system consistently delivered high accuracy across all three text

classification tasks without manual labels. Moreover, compared to directly using the GPT model for text classification, the second system achieved over 93% of its classification performance while requiring only approximately 6% of the associated time and financial cost.

Overall, this dissertation proposes cost-efficient and accessible solutions for text classification without relying on manually labeled data. By integrating large language models, self-supervised learning, active learning, and unsupervised learning, the proposed systems demonstrate the feasibility of building high-performance classification systems under limited supervision and resource-constrained conditions. The experimental results validate not only their technical effectiveness and generalization capabilities but also their practical value in scenarios where manual annotation is costly or infeasible. These findings contribute toward advancing more sustainable and adaptable machine-learning-based text classification technologies across diverse application scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Importance of text classification

Text classification is the conceptual task of assigning sentences, documents, or message streams to well-defined semantic categories, thereby imposing order on otherwise unstructured corpora that decision-makers can query, monitor, and reason over [1, 2].

Automatic text classification operationalizes this task with computational models that learn category boundaries from labeled examples, replacing time-consuming manual judgment with scalable algorithms. Today, machine learning-based automatic text classification systems underpin services such as information mining, spam filtering, and topic routing, reducing manual workload, enhancing information access, and improving analytical fidelity. Pioneering studies have shown that machine learning-based automatic text classification consistently reduces manual effort, accelerates information retrieval, and improves analytical abilities across application scenarios [1–5].

Text classification of software user reviews enables software develop-

ment teams to turn large amounts of feedback into clear, practical tasks. Software user reviews frequently embed maintenance-relevant feedback such as bug reports and concrete feature requests [6,7]. Employing text-classification techniques grounded in software engineering taxonomies can automatically separate these signals from noise, organizing them into structured categories that streamline triage, backlog prioritization, and the overall development workflow, thereby enhancing both process efficiency and product quality [7,8]. Empirical studies in application user feedback analysis further show that structuring reviews around concrete maintenance categories aligns backlogs with real user issues and feature requests, thereby improving software quality and customer satisfaction [9,10].

Sentiment classification transforms the large volume of online customer feedback into concise metrics that businesses can monitor to gauge perceived product and service quality. Positive review polarity is consistently linked to higher sales and stronger demand, while negative sentiment can depress revenue and alter market performance [11–13]. Tracking shifts in sentiment over time provides an early indicator of emerging issues and brand-health trends, enabling firms to prioritize fixes, adjust messaging, and address potential reputational threats before they escalate [14,15].

Topic classification organizes expansive document streams into coherent thematic layers that can be searched, analyzed, and reused. The DARPA Topic Detection and Tracking program first showed that routing broadcast-news stories into topic bins accelerates the discovery and follow-up of emerging events, markedly improving recall in large-scale retrieval tasks [16]. Subsequent work in newsroom settings demonstrates

that accurate topic classification underpins diversity-driven news recommendations, helping editors surface less-seen themes [17]. Controlled studies in enterprise archives further report that pre-classifying documents eliminates a substantial share of irrelevant hits and shortens search sessions, confirming the value of thematic indexing for knowledge management [18]. Collectively, these capabilities establish topic classification as a core enabler of timely information delivery and knowledge-driven decision making.

Toxic-language classification supplies digital platforms with a scalable first line of defense against harassment and hate speech. By mapping the continuous flow of posts, chat messages, and service e-mails onto an "acceptable/abusive" scale, machine-learned filters can auto-hide or hold most harmful contents for further moderator review before they reach audiences, leaving moderators to focus on nuanced or disputed cases [19]. Field deployments show that such models approach aggregated human-rater judgments on community corpora, and their adoption lets volunteer and staff moderators reallocate effort from routine removals to broader community-management tasks [19,20]. Surveys of hate-speech detection note that reliable automatic filtering supports legislative compliance and brand-safety objectives, motivating its uptake across social networks, gaming lobbies, and help-desk channels [21–23]. Finally, regulatory frameworks such as the EU Digital Services Act explicitly require removal of harmful content, making toxic-language classification a compliance as well as a safety imperative [24].

1.2 Challenges in machine learning–based text classification

Despite the demonstrable benefits of machine learning-based text classification, classic supervised pipelines still hinge on voluminous, high-quality annotations, a prerequisite that remains notoriously expensive. Large corpus projects report that even trained annotators plateau at roughly 750 tokens per hour while incurring substantial labor costs [25,26]. Subsequent empirical studies confirm that noisy or inconsistent supervision propagates directly into model error; consequently, ensuring label fidelity often requires domain specialists, inflating budgets yet further [27–30]. From a software engineering for machine learning standpoint, industry case studies likewise identify dataset construction as the dominant cost driver and emphasize the need for label-efficient workflows [31].

Rising model depth and complexity introduce further barriers such as domain expertise and computational resources. Hyper-parameter tuning experiments reveal acute performance sensitivity to configuration choices, which non-experts rarely optimize without protracted trial–and-error cycles [32]. Field reports from real-world deployments list hardware demands, maintenance overheads, and large-scale data management among the most persistent obstacles [33]. Moreover, training latency, computational resource consumption, and maintenance complexity negatively impact the continuous-integration/continuous-deployment (CI/CD) cycle [34,35]. Over time, these factors accumulate as "technical debt", magnifying coupling and brittleness in production systems [36]. Further, opaque models complicate troubleshooting and erode stakeholder trust, prompting calls for inherently interpretable alternatives or human-centred ex-

planation tooling [37, 38]. These intertwined data acquisition and engineering challenges serve as the central motivation for the cost-efficient systems developed and evaluated in this dissertation.

1.3 Research goal

The overarching objective of this dissertation is to design cost-efficient text classification systems that lower the overall "cost" while preserving strong text classification performance. Here, lower "cost" refers to building "accessible" text classification systems—by reducing annotation cost and minimizing the engineering challenges mentioned earlier, including the reliance on domain expertise and computational resources—in order to enable a wider audience to benefit from advances in text classification technologies. I decompose the notion of cost-efficiency into the following "accessible" yet high-performance goals:

1. **Primary target — reduce labeling cost:**

   Minimize the volume of manually labeled data required while still achieving high text classification performance.

2. **Objectives beyond reducing labeling cost:**

   Building on a low-labeling-cost, high-performance goal, I further aim to lower the engineering barriers, including domain-expert and computational resources demands, while maintaining high text classification performance:

   - Maintain a separate, decoupled, non-deep classification layer to reduce system coupling and potential technical-debt risks, while ensuring interpretability and maintainability.

- Ensure an efficient training process under resource-constrained settings, thereby reducing computational resource usage and enabling rapid model rebuilds that improve CI/CD readiness.

1.4 Contributions

The following contributions position this dissertation at the intersection of machine learning-based text classification and software engineering for machine learning.

1. This dissertation identifies the key challenges faced by machine learning-based text classification systems from a software engineering for machine learning perspective—including labeling cost as well as engineering challenges such as the need for domain expertise and substantial computational resources—and proposes two cost-efficient solutions.

2. Specific contributions of system 1, a cost-efficient software engineering-oriented game user reviews classification system.

   (a) A novel, cost-efficient text classification system is proposed under the paradigm of minimal supervision. It integrates self-supervised learning, unsupervised learning, and active learning to enable high-performing, software-engineering-oriented classification of game user reviews with minimal labeled instances. Under the same number of labeled instances, the proposed system achieves significantly higher performance than baseline approaches.

(b) Comprehensive cost-efficiency-oriented experiments further demonstrate that the proposed system achieves superior cost-efficiency in terms of both annotation cost and training time.

(c) This study targets the largely untapped yet significant game industry. To the best of my knowledge, this is the first machine learning-based work on game user review classification that adopts a software engineering–oriented perspective.

(d) A cross-game-type experimental dataset is constructed to ensure the generalization potential of the proposed system within the game review domain.

3. Specific contributions of system 2, a cost-efficient LLM-driven multi-strategy active learning system for cross-task text classification

(a) Proposed a cost-efficient, LLM-driven, multi-strategy active learning framework for cross-task text classification that eliminates the need for human-annotated data.

(b) The framework provides the first systematic comparison of diverse query strategies in an LLM-driven active learning setting, analyzing the effects of different seed and query batch pipelines on system performance.

(c) Through systematic studies on multiple datasets, it is demonstrated that under my system design, the proposed LLM-driven active learning pipeline achieves performance competitive with traditional manual active learning, while substantially reducing labeling costs.

(d) It presents a comprehensive comparison of monetary and time costs between my system and direct zero-shot LLM classifica-

tion, confirming the former as a practical alternative in resource-constrained scenarios.

(e) A structured prompt design is proposed to support multiple text classification tasks, achieving robust results across sentiment analysis, toxic comment classification, and topic classification datasets.

1.5 Outline of this dissertation

The remainder of this dissertation is organized as follows:

- Chapter 2 reviews the related work. It first introduces the foundational technologies used in this dissertation, including active learning, bidirectional transformer-based models, and generative pre-trained transformers. Then, it positions the two proposed systems in the context of prior work.
- Chapter 3 introduces the SE-SUA system—a cost-efficient software engineering-oriented text classification system for game user review classification. This system combines self-supervised learning, unsupervised learning, and active learning to achieve high accuracy with minimal labeled data. The chapter presents the system's architecture, implementation, and evaluation, and demonstrates the superior cost-effectiveness of the SE-SUA system.
- Chapter 4 presents a fully automated, large language model (LLM)-driven multi-strategy active learning system for cross-task text classification. The chapter evaluates four query strategies across three

different classification tasks—sentiment analysis, news categorization, and toxic comment detection—and demonstrates that, compared to directly using GPT for zero-shot classification, the proposed system achieves approximately 93% of its performance while using only about 6% of the time and monetary cost. Furthermore, comprehensive ablation studies confirm that the proposed LLM-driven active learning pipeline achieves performance competitive with traditional human-supervised active learning while substantially reducing annotation costs.

- Chapter 5 concludes the dissertation by summarizing the key findings, discussing broader implications for cost-efficient machine learning, and outlining future research directions. These include expanding to multilingual settings, improving model sustainability, and advancing real-world deployment to support broader technological democratization.

Chapter 2

Related work

Building upon the challenges of ML-based text classification systems analyzed from the perspective of software engineering for machine learning in Chapter 1, as well as the goals derived from these challenges, this chapter will first introduce the relevant technologies employed in the proposed systems, followed by a detailed positioning of the two proposed systems within the context of existing work.

2.1 Relevant technologies

2.1.1 Active learning

Active learning has been widely studied as an effective approach to reduce labeling costs by selecting the most informative samples for annotation. Angluin [39] first formalized the concept of membership and equivalence queries, showing that by selectively querying an oracle, a learner could achieve exponential reductions in sample complexity in certain settings. Later, Cohn et al. [40] bridged this theory to practical algorithms

by proposing selective sampling techniques that leveraged model uncertainty to identify informative examples. Since then, a wide range of query strategies have been proposed under this general framework, including entropy-based Query-by-Committee (QBC) [41–43], diversity-aware entropy sampling [44], core-set sampling [45], and information density sampling [46], each aiming to optimize label acquisition for supervised learning.

The active learning component of system 1 adopts the entropy-based Query-by-Committee (QBC) method, which has been extensively studied and empirically validated as an effective approach for text classification tasks [47–50]. Building upon this foundation, a clustering-based seed set selection method is proposed to improve the efficiency of initial labeling.

On the other hand, system 2 fills the gap in the lack of comprehensive comparisons of different query strategies within the LLM-driven active learning framework (a specific form of active learning, to be elaborated in Section 2.3). It conducts a comprehensive analysis of the performance of four distinct query strategies—QBC, entropy sampling with diversity, core-set sampling, and information sampling—within the LLM-driven active learning framework.

2.1.2 Bidirectional transformer-based model

Devlin et al. [51] introduced BERT (Bidirectional Encoder Representations from Transformers), which employs a bidirectional attention mechanism to capture contextual information from both preceding and following tokens. This bidirectional modeling significantly improves the quality of learned text embeddings, making them more suitable for a wide range of

downstream tasks. Based on BERT, Liu et al. [52] introduced RoBERTa, which removes the Next Sentence Prediction (NSP) objective and incorporates dynamic masking, larger batch sizes, and extended pretraining to enhance pretraining effectiveness.

While BERT and its variants can be fine-tuned for text classification tasks, studies have revealed that fine-tuning on small labeled datasets leads to unstable performance. Dodge et al. [53] empirically demonstrated that when fewer than 1,000 labeled examples are available, the performance of fine-tuning becomes highly sensitive to random seeds. Mosbach et al. [54] also confirmed this instability and attributed much of it to inappropriate learning rate schedules. Collectively, these studies highlight that fine-tuning BERT and its variants with limited labeled data is unstable. The proposed mitigation strategies, given in these studies—such as repeated restarts, extensive hyperparameter search, and longer training—conflict with the research goals introduced in Chapter 1 from the software engineering for machine learning perspective, which aim to reduce engineering challenges.

Given these limitations, and in alignment with the design principles proposed in Chapter 1—specifically, the separation of a non-deep, decoupled classification layer to improve interpretability, maintainability, and reduce coupling—system 1 and system 2 use BERT and RoBERTa as embedders. Instead of end-to-end fine-tuning, both systems leverage these bidirectional transformer-based models to generate contextual representations. Classification is performed in a decoupled downstream module, specifically a logistic regression classifier trained under an active learning framework. This design enables learning from limited labeled data while addressing the engineering challenges identified in Chapter 1.

2.1.3 Generative pre-trained transformer

Brown et al. [55] introduced GPT-3, a state-of-the-art language model capable of performing various NLP tasks such as text completion, translation, and summarization, with minimal or no task-specific training data. This model exemplifies zero-shot learning capabilities, where extensive pretraining on diverse datasets allows it to generalize across different tasks by leveraging its broad knowledge base rather than explicit supervision.

Building upon GPT-3, OpenAI released GPT-4, a multimodal large language model capable of processing both textual and visual inputs [56]. The model demonstrated strong performance across a broad range of tasks, including complex reasoning, multilingual understanding, and standardized test benchmarks, establishing a new level of general-purpose language intelligence. More recently, GPT-4o was introduced as an optimized version of GPT-4 with enhanced multimodal capabilities and real-time interaction features [57]. It supports synchronous text, vision, and audio inputs and outputs, enabling fluid and cost-efficient user experiences in interactive applications. These developments illustrate the continuing trend of advancing LLMs toward more general, real-time, and multimodal AI systems.

Alongside these advancements, generative AI technologies have been adopted in a range of practical applications. Notable examples include open-domain question answering [58], news summarization [59], and code generation [60], where generative models support complex reasoning, abstraction, and content creation. These systems are also increasingly integrated into real-world workflows, such as virtual assistants and pro-

gramming support tools, reflecting the growing presence of generative models in diverse AI-driven applications.

While generative AI models like GPT have demonstrated impressive performance across multiple domains, their application in structured classification tasks at scale faces certain limitations and challenges. Rae et al. [61] analyzed the computational trade-offs of scaling large language models, highlighting that while larger models achieve improved performance, they also introduce inefficiencies in inference and deployment. The study shows that GPT-style models, due to their autoregressive nature, require sequential token generation, limiting parallelization and increasing inference latency. Moreover, Raiaan et al. [62] provided a comprehensive analysis of large language models (LLMs), highlighting both their advancements and inherent challenges. Their study emphasizes that while models like GPT demonstrate strong performance across various NLP tasks, their deployment at scale faces computational and efficiency constraints. In addition to discussing the limitations posed by the autoregressive decoding mechanism, they further elaborate on the challenges introduced by high computational cost and energy consumption in real-world applications.

To address these challenges, system 2 integrates a generative pre-trained transformer with an active learning framework—referred to as LLM-driven active learning—to mitigate the inefficiencies of directly using GPT models for text classification tasks. The positioning of system 2 within this line of research is detailed in Section 2.3.

2.2 Positioning of system 1: cost-efficient software engineering-oriented game user reviews classification system

2.2.1 Information contained in user reviews

Many researchers have conducted various studies on the information contained in user reviews that can help in software development.

Harman et al. [63] analyzed the difference between online and offline software sales models. They pointed out that app marketplaces are a new form of software repositories, distinct from traditional app marketplaces. They introduced app store mining and analysis as a form of software repository mining. Their findings demonstrate the value of app store analytics. Their study also found that information from user reviews on online sales platforms can be used to help developers improve their products.

Palomba et al. [10] point out that the online sales model necessitates a much shorter update cycle for software development companies, and that software development teams need to find informative reviews and update software based on user feedback in a shorter period of time. They also have pointed out that user reviews contain high-value information that can make the software more likely to succeed.

Hassan [64] explores information that can be mined from many sources, including emails, change logs, configuration files, user documentation, bug reporting systems, and source code. Their research revealed that the data contained a wealth of information about the systems under investigation.

Pagano and Maalej [9] conducted an empirical study to analyze app store user reviews information to help developers. Their work concludes that user feedback and user involvement are crucial for modern software organizations.

2.2.2 Classifying and analyzing user reviews

Numerous studies have investigated the analysis and classification of user reviews.

Guzman et al. [65] presented a taxonomy for ordering app reviews into categories that are relevant for software evolution. They performed an experiment to compare the performance of different machine learning approaches into the categories defined in their taxonomy. They combine TF-IDF with Naive Bayes, SVM, Logistic Regression, and Neural Network classifiers. The results for the classification of the reviews are encouraging. They conclude that the software developers could use their classifier for analyzing user reviews and prioritizing their tasks. They believe that their approach can be extended to automatically rank the categorized reviews by taking ratings and sentiments contained in the reviews into consideration. In this way, reviews with a more negative sentiment or rating could be given a higher priority. Additionally, mechanisms to summarize and visualize the content of the classified reviews could further reduce the processing effort.

Rustam et al. [66] stated that app stores usually allow users to give reviews and ratings to the apps in the market. Such information can be used by the developers to resolve issues and make further plans for future updates. In order to utilize this information, they stated that user reviews

need to be classified into happy and unhappy. To do so, they used BoW, TF-IDF, and CHI2 combined with several different classifiers to classify reviews for Shopify apps into happy and unhappy.

Lu et al. [67] stated that the leading app distribution platforms contain a very large number of user reviews. Research shows that user reviews contain abundant useful information that may help developers improve their apps. Thus, they aim to extract and consider the non-functional requirements that describe a set of quality attributes wanted for an app and are hidden in user reviews on the app distribution store. They combined BoW, TF-IDF, CHI2, and classifiers such as Naive Bayes, Bagging, and J48 to classify user reviews into four types of NFRs. Their results show that the combination of augmented BoW with Bagging achieves the best result.

Carreno et al. [68] stated that user feedback is imperative in improving software quality. They explore the rich set of user feedback available for third-party mobile applications as a way to extract new/changed requirements for next versions. They think that a potential problem using user review data is its volume and the time commitment involved in extracting new/changed requirements. Thus, they proposed a new approach to alleviating part of the process through automatic topic extraction. They process user reviews to extract the main topics mentioned as well as some sentences representative of those topics. They stated that this information can be useful for requirements engineers to revise the requirements for the next releases. And their results show that the automatically extracted topics matched with the manually extracted topics.

Chen et al. [69] believe that the app market is growing bigger and bigger. App developers spend considerable effort on collecting and ex-

ploiting user feedback to improve user satisfaction, but suffer from the absence of effective user review analytics tools. Thus, they proposed AR-Miner, a mobile app review mining framework to facilitate app developers to extract the most "informative" information from raw user reviews in app marketplaces with minimal human effort.

Research dedicated specifically to game user reviews remains largely limited to sentiment polarities. Viggiato et al. [70] evaluated three off-the-shelf sentiment classifiers (NLTK Naïve Bayes, Stanford CoreNLP, and SentiStrength) on game user reviews, reporting that the best model, NLTK Naïve Bayes, achieved 0.61 accuracy and 0.70 AUC; their error analysis traced misclassifications to mixed pros–cons sentences, domain-specific slang, and sarcasm. Tan et al. [71] compared five text classification algorithms (SVC, MLP, XGBoost, Logistic Regression, Multinomial NB) on a three-way polarity task using reviews from Steam and Metacritic, finding that a TF–IDF + SVC configuration outperformed all alternatives, reaching 0.91 accuracy. While these studies demonstrate that machine learning-based classification pipelines can effectively capture sentiments in game reviews, they fall short in linking user feedback to actionable software maintenance categories and do not consider annotation cost or cost-efficiency—limitations that the proposed system 1 is designed to address.

From a broader software engineering-oriented user review classification viewpoint, Dabrowski et al. [72] conducted a systematic review of research on application review analysis. They highlighted three persistent shortcomings: (i) the heavy workload associated with manual labeling, calling for methods that minimize annotation effort; (ii) scant attention to efficiency and scalability metrics, with existing studies reporting only

predictive effectiveness; and (iii) limited evidence of practical applicability in production environments. These conclusions underscore the need for cost-sensitive review-classification approaches, which directly motivate the SE-SUA system presented in Chapter 3.

2.3 Positioning of system 2: cost-efficient LLM-driven multi-strategy active learning system

With the advancement of large language models (LLMs), recent work has begun to explore the integration of active learning with LLMs. The first line of work keeps humans in the annotation loop while allowing a large language model to drive the query stage. Margatina et al. [73] recast in-context demonstration selection as a single round, pool-based active learning problem and tested semantic similarity, diversity, and GPT-perplexity uncertainty sampling, across twenty-four classification and multiple-choice datasets. Their experiments show that similarity sampling consistently outperforms the other criteria, yet every acquired instance still requires human labeling, and the one-off acquisition step falls short of a full iterative active-learning cycle. Diao et al. [74] extend this idea to Chain-of-Thought prompting. Their Active-Prompt framework first ranks candidate questions by predictive disagreement produced by an LLM and then asks human annotators to write CoT rationales for the most uncertain items. Experiments on eight reasoning benchmarks demonstrate that combining LLM-estimated uncertainty with human-written reasoning yields high-quality few-shot exemplars, but the method remains human-dependent and is aimed at constructing demonstrations rather than automating the active learning loop.

The second line of research leans more toward the pseudo-labeling paradigm. For example, Xiao et al. [75] treat the LLM as a coarse annotator that pseudo-labels every example, and then introduce a small language model (SLM) that learns from these noisy labels and feeds back high-confidence instances for the LLM to relabel. This method falls under the scope of pseudo-labeling, where relabeling is used to improve annotation quality. Furthermore, it is non-incremental, requiring the entire dataset to be labeled upfront, without any dynamic control over annotation cost.

In contrast, active learning aims to maximize model performance per unit of labeling cost by employing sampling strategies that prioritize only the most informative instances for labeling. Following this paradigm, more relevant works include the study by Zhang et al. [76], which initializes uncertainty and diversity-based query strategies with human-annotated gold data and then uses GPT to annotate the subsequently selected instances for named entity recognition and relation extraction tasks. More recently, Rouzegar and Makrehchi [77] proposed an active learning pipeline combining uncertainty sampling with GPT-3.5-based annotation. In their framework, human annotators act as a fallback for instances where the automatic labeling is uncertain. Their experiments across multiple text classification tasks show that such hybrid annotation can reduce labeling costs while maintaining performance. However, their query strategy is limited to classic uncertainty sampling, and their fully automated pipeline did not achieve performance competitive with traditional human-annotated active learning. Although the hybrid setting reduces annotation costs, it still relies on rather costly human labeling as a backup mechanism.

In summary, current attempts to integrate active learning with large language models (LLMs) either use LLMs to simulate the query strategy component of active learning—while still relying on human annotation—or fall within the scope of pseudo-labeling approaches, which lack incremental control over annotation cost. Among the limited studies that are more aligned with my work—namely, those aiming to replace the human oracle in the active learning process with an LLM—human annotations are still retained at critical points. Furthermore, these studies often overlook key aspects extensively studied in traditional active learning, such as the impact of different query strategies and initialization methods.

Motivated by these gaps, Chapter 4 introduces a cost-efficient and fully automated multi-strategy active learning framework, in which a structured prompting mechanism is designed to support a variety of text classification tasks. I systematically compare different annotation strategies for both the seed set and queried instances, and conduct a cost-efficiency-oriented analysis in terms of both time and monetary cost against directly applying GPT-based zero-shot classification. Moreover, through extensive experiments, I find that under my system design, the proposed LLM-driven active learning pipeline achieves performance competitive with traditional manually supervised active learning, while substantially reducing annotation costs.

Chapter 3

Cost-efficient software engineering-oriented game user reviews classification system

3.1 Introduction

Building on the motivation for cost-efficient text classification, this chapter turns to the game industry, an industry characterized by a large user base and a rapid update cycle, which together increase the demand for automated text classification. Steam, one of the largest digital game stores, recorded 120 million monthly and 62.6 million daily active users in 2020 [78]. A headline release can accumulate more than 150,000 reviews within its first week on the platform [79]. These large volumes of feedback frequently include production bugs, performance bottlenecks, and missing features that teams must triage before ratings and revenue decline. The complexity of modern game software has risen sharply, and hot-fix mechanisms allow studios to publish patches within hours of defect discovery [80, 81]. This rapid update cycle forces developers to mon-

itor reviews continuously or else the high number of negative sentiment will damage sales [67]. Beyond defect detection, mined reviews help engineers understand evolving user needs and prioritise maintenance [9]. Yet manual inspection is infeasible at this scale, and conventional supervised pipelines demand large, high-quality corpora—an effort long recognised as labour-intensive and costly [25, 26]. Furthermore, existing machine-learning research on game reviews is largely confined to coarse sentiment labels [70, 71], and Dabrowski et al. [72] highlight the open need to lower annotation effort while evaluating efficiency and scalability alongside accuracy in the software user review classification field.

Against this backdrop, I propose SE-SUA (a **S**oftware-**E**ngineering-oriented system that combines **S**elf-supervised, **U**nsupervised, and **A**ctive learning), a cost-efficient, software-engineering-oriented system for classifying game user reviews. By combining these learning paradigms, SE-SUA minimizes manual labeling effort while jointly optimizing annotation cost, training time, and classification performance.

3.2 Proposed SE-SUA system

This section describes the design of the proposed system for cost-efficient game user review classification. The system is designed to reduce annotation costs and enable efficient training while maintaining high classification performance on game user reviews.

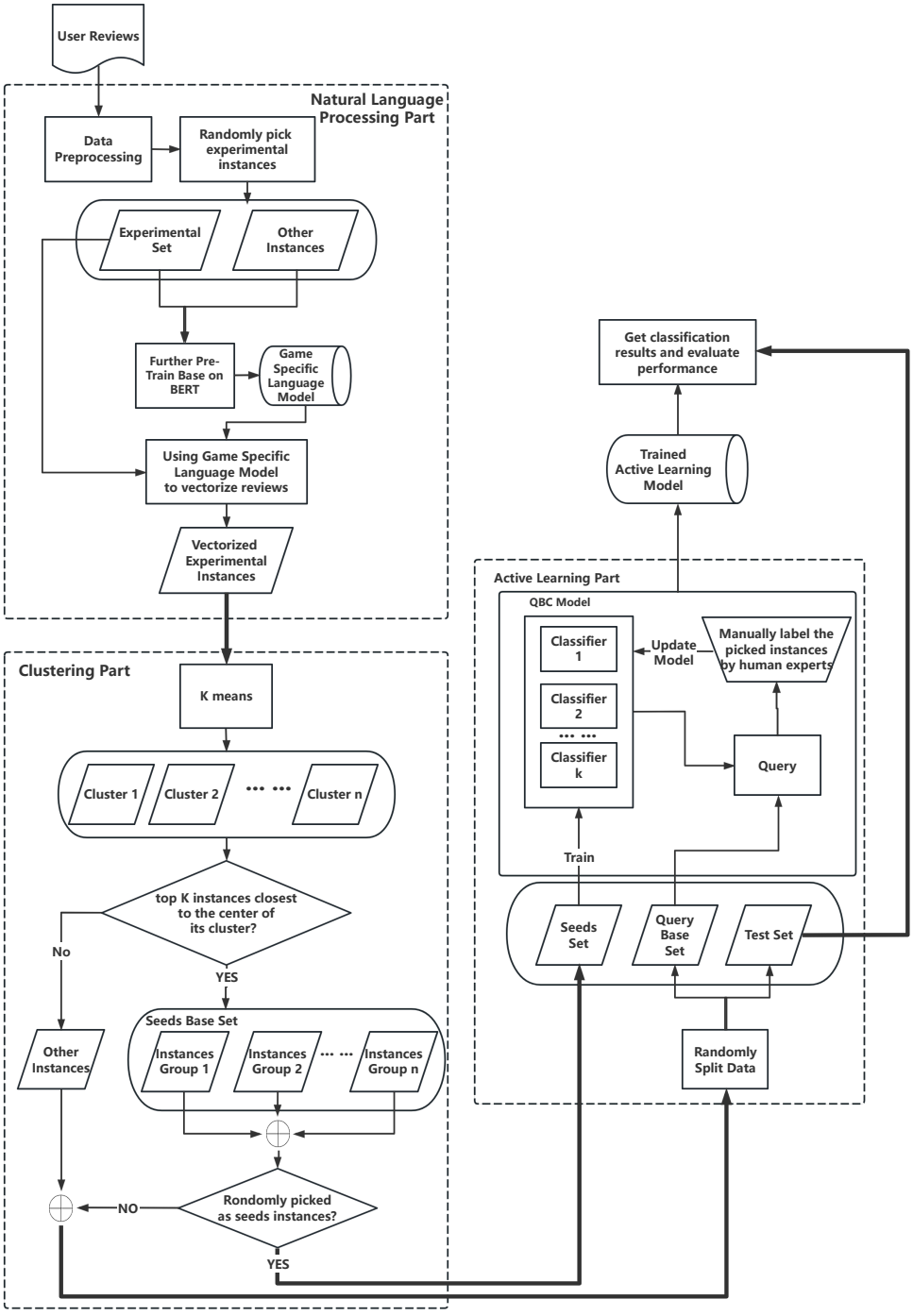

Figure 3.1: Overall workflow of my system

3.2.1 Overview

The key idea behind SE-SUA's cost-efficiency lies in its strategic integration of three learning paradigms. Self-supervised learning leverages abundant unlabeled data to construct robust language representations without incurring annotation costs. Unsupervised learning employs clustering to identify representative seed instances, thereby limiting initial labeling to a small subset and improving labeling efficiency. Active learning further improves cost-effectiveness by querying only the most informative instances for manual annotation. The overall workflow of my system is shown in Figure 3.1.

- The main function of the natural language processing part is to train a Game Specific Language Model and use it to convert user reviews into vectors, which will be described in detail in Section 3.2.2.
- The main function of the clustering part is to get the initial labeled instances (seed instances) needed to start active learning. Section 3.2.3 introduces the detailed design of this part.
- The main function of the active learning part is to train a classifier to classify user reviews with a small number of labeled instances, and the detailed design of this part will be introduced in detail in Section 3.2.4.

3.2.2 Natural language processing design

3.2.2.1 Overview

The main task of this part of my system is to obtain the vectorized game reviews for further processing in the clustering and active learning parts. This section will focus on introducing the design of language modeling and text vectorization.

- Subsection 3.2.2.2 introduces the foundation and rationale behind my Game-Specific Language Model training.
- Subsection 3.2.2.3 describes how I further pre-train BERT.
- Subsection 3.2.2.4 introduces the design of how to vectorize the text.

3.2.2.2 Foundation and rationale of the Game Specific Language Model

To minimize annotation costs while obtaining high-quality textual representations, I adopt BERT as the foundation of my system's language modeling component.

In previous studies, various methods have been used to vectorize user reviews. For example, Lu and Liang [67] explored the Bag-of-Words (BoW), Term Frequency - Inverse Document Frequency (TF-IDF), and Chi-Squared in their study. Rustam et al. [66] also discussed these models.

Natural language processing has advanced so that I can now apply newer techniques to my system. When I analyzed the problems faced, I found that compared with labeled instances that required human resources, I could easily obtain a large number of raw user reviews. My

idea is to use these unlabeled user reviews to help train my system. And this fits the problem that BERT solves.

Bidirectional Encoder Representations from Transformers (BERT) is a language representation model introduced by Devlin et al. [51]. Unlike previous language representation models, BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context.

BERT's Masked Language Modeling (MLM) training task is in line with the idea of self-supervised learning, which does not require any labeled instances. In MLM, a subset of input tokens is masked, and the model is trained to predict the original tokens at those positions using both left and right context. Consequently, MLM leverages raw text as supervision—the masked tokens serve as the targets—while learning bidirectional contextual representations. This enables BERT to better capture text semantics and contextual dependencies for downstream tasks.

3.2.2.3 Further pre-training

I chose to further pre-train BERT to obtain a language model more adapted to the game review text. The importance of further pre-training has been confirmed by Gururangan et al. [82]. Their work shows that further pre-training the model towards a specific task or small corpus can provide significant benefits. More importantly, their findings show that further pre-training is beneficial even for an extensively parameterized language model such as BERT and other language models based on BERT. In addition, since my system does not include downstream tasks such as question answering, I choose to use only the Masked Language Modeling to

further pre-train BERT.

Thus, in my system, by using Masked Language Modeling, I further pre-train the BERT base uncased model [51] with unlabeled user reviews that do not require manual labeling and are easy to obtain.

3.2.2.4 Vectorization

The pre-training results in the Game Specific Language Model, which is used to vectorize user reviews for further processing and classification. The workflow of the vectorization process is shown in Figure 3.2. In order to obtain a vector that can represent each user review's feature, first, the contextualized token embeddings output by the Game Specific Language Model are obtained. Then, a pooling layer is added. The function of the pooling layer is to transform all the token embeddings within a review into a vector that can represent the whole text feature by certain computational methods. For the computational method, I adopt mean pooling, which is reported as a robust solution in [83], that is, calculate the average of token embeddings as the vector of the whole review.

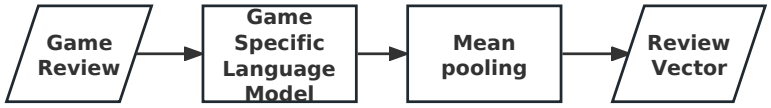

Figure 3.2: Vectorization process

3.2.3 Clustering design

3.2.3.1 Overall design

The overall procedure of the clustering part is as follows:

1. Cluster the vectorized instances (user reviews) passed by the natural language processing part.
2. Output the instances near the center of each cluster.
3. Form the initial labeled instances set (Seeds Set), which will be used to start the active learning process

The main task of the clustering part is to get the initial labeled instance set (Seeds Set) needed to start active learning. To be specific, before starting to run active learning, a small amount of initial labeled instances is needed to train the active learning model.

Instead of choosing these instances randomly, the proposed system clusters the user review vectors obtained by the NLP part, and then outputs user reviews close to the centroid of each cluster as representative reviews, which will be put into the Seeds Base Set. The process of constructing the Seed Base Set is formalised in Formula (3.1). Given a partition of the unlabeled pool into K clusters $C_1, C_2, \ldots, C_K$ obtained with k-means, the goal is to select N highly representative instances from each cluster and merge them into the Seed Base Set B:

$$B = \bigcup_{k=1}^{K} \underset{\substack{S \subset C_k \\ |S|=N}}{\arg\min} \sum_{x_i \in S} \|x_i - \boldsymbol{\mu}_k\|_2^2, \tag{3.1}$$

where $\boldsymbol{\mu}_k$ is the centroid of cluster C_k and $\|x_i - \boldsymbol{\mu}_k\|_2^2$ denotes the squared Euclidean distance between instance x_i and its centroid. For each cluster, the algorithm chooses the subset $S \subset C_k$ of cardinality N whose members minimise the total squared distance to $\boldsymbol{\mu}_k$; the union of these K subsets constitutes the complete Seed Base Set B.

This clustering part will help users of my system to limit the initial labeling to a small number of instances in a more purposeful and efficient way by only presenting representative reviews instead of a massive amount of instances including noise. Also, in this step, users of my system can customize their own classification standards according to their own product line and development team's structure.

3.2.3.2 K-means

I use K-means for clustering (Figure 3.3). The algorithm begins by randomly selecting K instances from the input data as the initial centroids. Then, it calculates the distance between each instance and the current centroids. Based on the minimum distance, each instance is assigned to its nearest centroid, forming K groups. The algorithm then recalculates the centroids by taking the average of the instances in each group. This process repeats: distances are recalculated, assignments are updated, and centroids are recomputed, until the centroid positions no longer change significantly. This iterative process ensures convergence to a stable clustering solution.

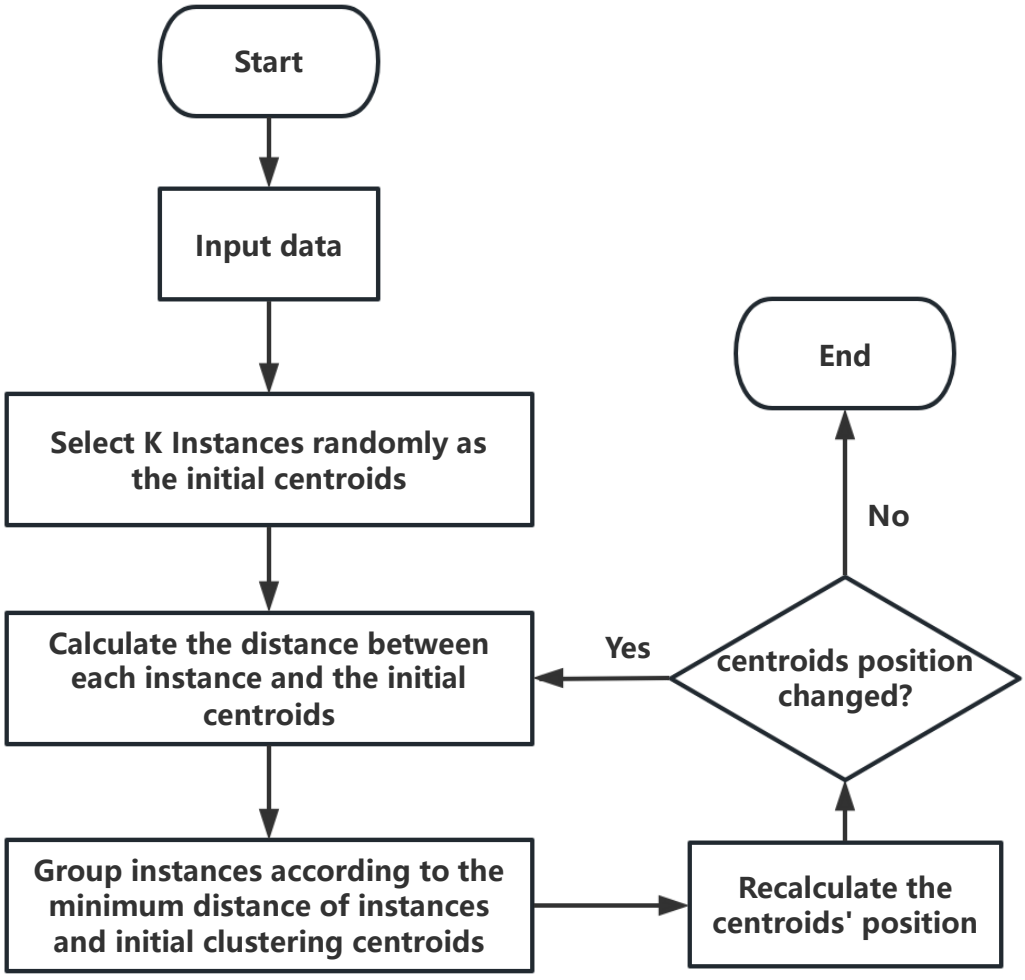

Figure 3.3: Workflow of K-means

3.2.4 Active learning design

3.2.4.1 Overview

The active learning component of my system is built upon the Query-by-Committee (QBC) [43] query strategy, combined with the proposed clustering-based seed selection method. The overall procedure is as follows:

1. Seed instances (passed from the clustering part) are used to form a "committee".
2. The committee goes through the unlabeled instances and chooses instances to query.
3. Human experts give labels to the queried instances.
4. Newly labeled instances will be added to the training set to update the model.

3.2.4.2 Query by committee active learning

Generally speaking, the training of machine learning models requires a large amount of manually labeled training instances, and models deployed in the industrial field usually require new labeled instances continuously. On the other hand, there is a phenomenon of diminishing marginal returns for the model on the data side. More data does not always bring continuous linear performance improvement. At the same time, considering the time and economic cost of manual labeling, the reduction of manpower during labeling while obtaining benefits has be-

come an important issue, and active learning has emerged to find that balance.

According to Settles [84], there are machine learning problems in which unlabeled instances may be abundant, but labeling is difficult, time-consuming, and expensive. Active learning helps to complete these tasks at a lower cost by actively selecting the most valuable samples and handing them over to human experts for labeling.

To be specific, the main task of the active learning part of my system is to train an accurate prediction model by using a small number of labeled instances to reduce the need for manpower on labeling instances. Only the most valuable instances will be queried. The learning cycle in active learning is shown in Figure 3.4 [84].

My system uses Query-By-Committee (QBC) [43] query strategy to find the most valuable instances to which humans will add labels. QBC [43] is a sampling strategy based on version space reduction, and the core idea is to preferentially select unlabeled samples that can reduce the version space to the greatest extent. QBC consists of two basic steps:

1. Using multiple classifiers to form a committee.
2. All models in the committee make predictions on unlabeled samples in turn and prioritize the most inconsistent voting samples for labeling.

3.2.4.3 Entropy based query by committee

Due to the fact that QBC needs to train several models in the process of practical application, it has high computational complexity. Based on

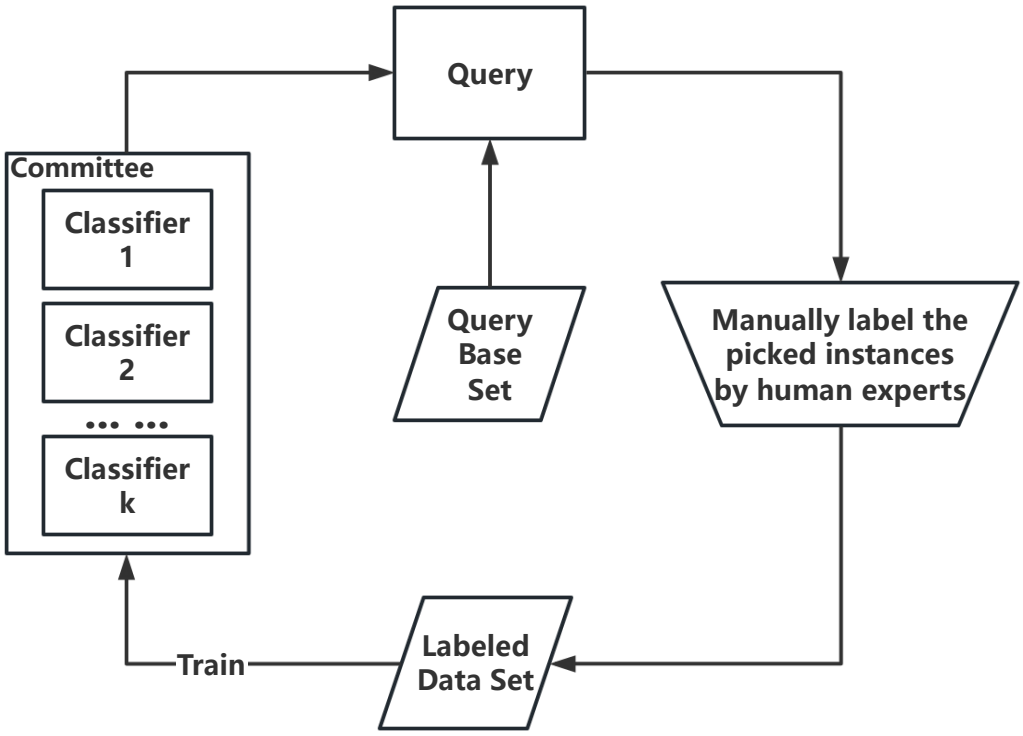

Figure 3.4: Learning cycle of active learning

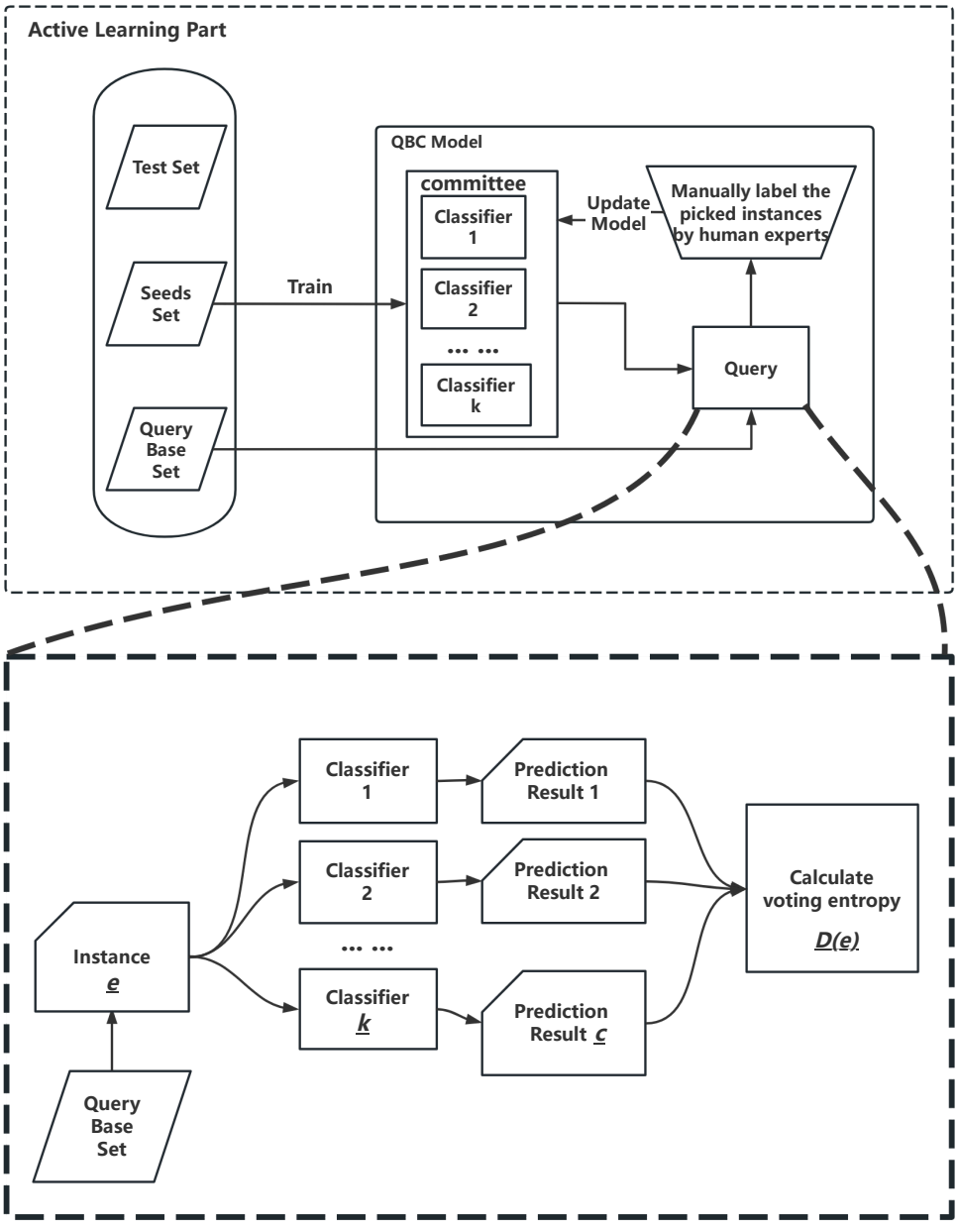

Figure 3.5: Workflow of calculating voting entropy

this, the Entropy based Query-By-Bagging introduces the bagging inheritance method and bootstrap sampling to solve the high computational complexity problem [41] [42]. In my system, I adopt the QBC strategy using the Entropy based QBC. In Entropy based QBC, the basis for judging "most inconsistent" is voting entropy as shown in Figure 3.5, which illustrates the workflow for selecting query instances, detailed as follows:

1. Train k classifiers to form the committee.
2. Classify each instance e with each classifier in the committee.
3. Get the classification results c_i of each classifier.
4. Calculate disagreement D_e according to c_i
5. Select the instances to query based on D_e
6. Human experts give labels to the queried instances.
7. Newly labeled instances will be added to the training set to update the model.

Within the above process, the disagreement of different classifiers in the committee is measured by voting entropy D_e. Voting entropy is a natural measure for quantifying the uniformity of classes assigned to an example by the different committee members. It can be further normalized by a bound on its maximum possible value $\log \min(k, |C|)$, giving a value between 0 and 1. Denoting the number of committee members assigning a class c for input instances by $V(c, e)$, the normalized voting entropy is shown in Formula (3.2) [85]:

$$D(e) = -\frac{1}{\log \min(k, |C|)} \sum_{c} \frac{V(c,e)}{k} \log \frac{V(c,e)}{k} \tag{3.2}$$

- $|C|$ denotes the total number of categories
- k denotes the number of classifiers in the committee
- V (c, e) denotes the number of committee members assigning a class c for input instances

The normalized voting entropy is 0 when all classifiers in the committee make the same prediction, which indicates that the class of this instance is almost certainly the predicted class for the current classification model. In this case, adding this sample to the training set can provide little help in improving the model. In contrast, the greater the prediction of sample labels differs between the classifiers in the committee, the greater the voting entropy is, and the greater the amount of information provided by this sample can help improve the prediction accuracy.

After the QBC strategy chooses the instances with high voting entropy values, the human experts label these instances. After that, the newly labeled instances will be passed to the training data set to update the active learning model.

3.2.4.4 Conclusion

After completing the active learning, a classifier can be trained using the selected instances, which will be used for game user reviews classification. Moreover, during the training process, users of my system only need to give a small number of labels when the query strategy issues a query.

In the entire system, only a small amount of labeled instances is needed in the training and iterative active learning process and the clustering part. My entire system tries to find the most efficient way to use and add labeled instances to complete the classification of game user reviews.

3.3 Implementation

This section specifies the process of implementing the proposed system, which contains three main parts. This section is organized as follows:

- Section 3.3.1 describes in detail the process of data preparation.
- Section 3.3.2 details the implementation of the natural language processing part.
- Section 3.3.3 describes the implementation for the clustering part.
- Section 3.3.4 explains the implementation of the active learning part.

3.3.1 Data preparation

The data processing stage mainly includes the following steps:

1. Crawl and preprocess game user reviews.
2. Randomly select experimental instances from the preprocessed user reviews.
3. Manually label the experimental instances for subsequent experiments and result evaluation.

The first step is to crawl user reviews from STEAM. As mentioned in Section 3.1, the STEAM platform is currently the mainstream online game sales platform, with a large number of users and user reviews. Figure 3.6 gives two examples of user reviews on STEAM. Since the main goal of this study is to help game developers efficiently analyze user feedback to obtain information about game defects and fix bugs or update the game accordingly, I decided to select only those user reviews that gave negative (thumbs down) ratings.

The STEAM platform divides games into six categories. In order to ensure the diversification of data, I crawled 2,000 negative user reviews from each category on the STEAM platform. This resulted in a total of 12,000 negative user reviews. An example of a crawled user review is shown in Table 3.1. The rating (Recommended/Not-recommended) and the review body text are crawled. The body text part will be used in my system, and the rating part is only used to double-check that I only crawled negative reviews. Then, as shown in Figure 3.7, I removed the non-English reviews and removed all symbols except " ' . ! ? ".

After data preprocessing, I randomly picked 10,000 preprocessed user reviews to form the Preprocessed Set, of which 3,000 user reviews were randomly selected as the Experimental Set. These experimental instances are manually labeled for subsequent experiments and evaluation. The labeling work is based on my classification factors, which were designed based on the idea of sending user reviews as feedback to corresponding departments within game companies. Thus, I classify game reviews into Production Team Issues, Operations Team Issues, and Design Team Issues. The classification factors I use are given in Table 3.2, and are based on the work of Ramadan et al [86]. Other classification factors could theo-

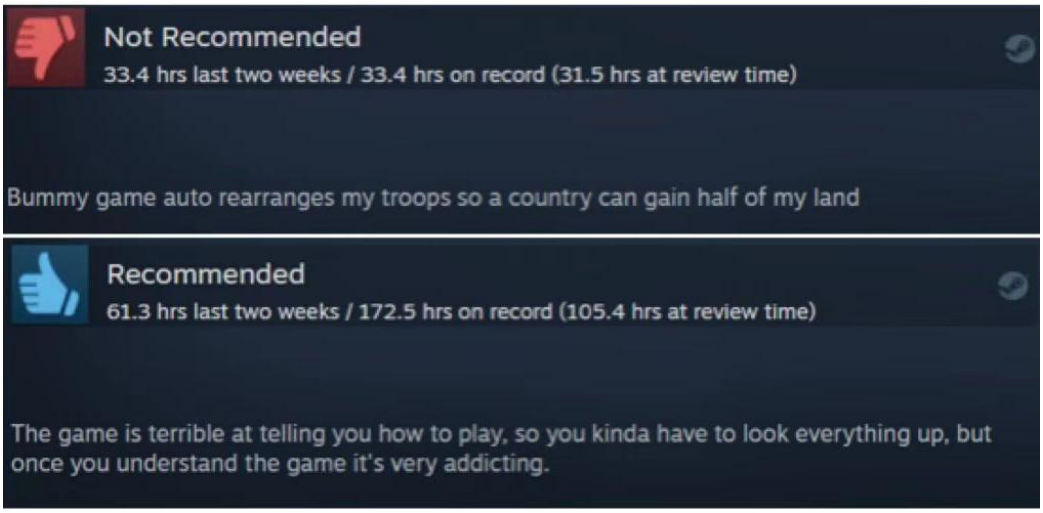

Figure 3.6: User review examples

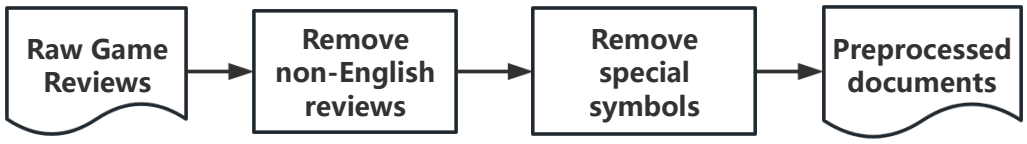

Figure 3.7: Data preprocess

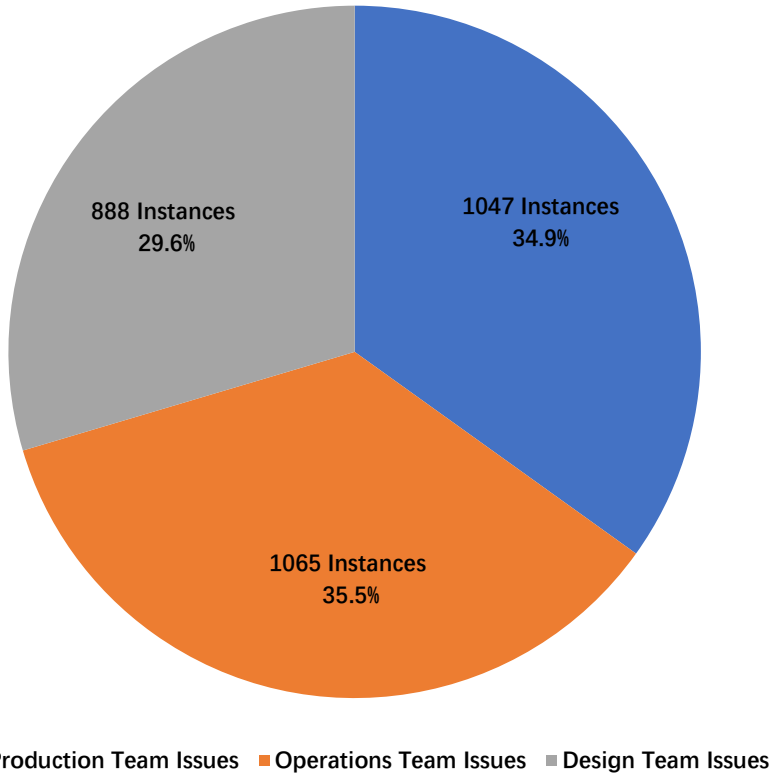

Figure 3.8: Distribution of experimental instances in each category

retically be used in my system. The distribution of experimental instances in each category are shown in Figure 3.8.

3.3.2 Implementation of natural language processing part

The overall procedure for implementing the natural language processing part is as follows:

1. Train BERT with preprocessed user reviews to obtain the Game Spe-

Table 3.1: An Example of a Crawled User Review

Key1: rating	**Key2: review body**
Not Recommended	The optimization in this game is terrible. There are moments where the action is so choppy I can't even tell what's going on. Sometimes my character will twitch and jump around for no reason and it's very difficult to target enemies accurately when the frame rate is all over the place. It really sucks the fun out of what should be an exciting experience. This issue needs to be fixed immediately.

cific Language Model.

2. Vectorize the experimental instances using Game Specific Language Model and mean pooling strategy.

3.3.2.1 Further pre-train BERT

The further pre-training is based on BERT-base-uncased model. BERT-base-uncased is a pre-trained model on English language [51].

I further pre-train the BERT-base-uncased model to obtain a new language model that specifically focuses on game reviews. I take all the instances in the Preprocessed Set and use the Masked Language Modeling method stated in Section 3.2. In order to ensure that my system's performance is not attributed to task-specific hyperparameter tuning, but rather to the core design of the overall pipeline, I retain the default configuration of BERT-base-uncased. This choice supports general applicability and fairness in comparison. Table 3.3 lists the parameter settings used

Table 3.2: Classification Factors

Class	**Description**	**Example**
Production Team Issues	1. Whether there are bugs that affect the game experience. 2. Whether the game's features work as designed. 3. Whether the optimization of the game is successful and can adapt to the hardware resources of most players.	Optimization is terrible. Your game will be stuttering all the time no matter if you try the lowest graphic settings.
Operations Team Issues	1. Whether the game's server is stable. 2. Whether the game is balanced, and whether any game props or external software affect the balance.	Laggy servers. PVP is god-awful. Only buy this if you enjoy the PVE aspect. Overall not worth 60.
Design Team Issues	1. Whether the game story is logically coherent or unfinished. 2. Whether the game content is complete or missing features. 3. Whether the mechanics/interface are intuitive and easy to understand.	Downgraded UI compared to CK2. Needs rework. Confusing to play in current state.

during pretraining.

Table 3.3: Language Modeling Parameters for Game Specific Language Model

Parameters	**Details**
Base Model	bert-base-uncased
hidden_size	768
max_position_embeddings	512
num_attention_heads	12
num_hidden_layers	12
transformers_version	4.19.0.dev0

- Hidden_size denotes the dimension of the encoder layers and the pooler layer.
- Max_position_embeddings denotes the maximum sequence length that this model might ever be used with.
- Num_attention_heads denotes the number of attention heads for each attention layer in the Transformer encoder.
- Num_hidden_layers denotes the number of hidden layers in the Transformer encoder.

3.3.2.2 Vectorize game reviews using Game Specific Language Model

This part introduces how I built the Game Reviews Vectorizer, which is used to vectorize the user reviews.

The vectorizer consists of the Game Specific Language Model embedder and the mean pooling method. The embedder transforms each token into a 768-dimensional vector. Then, as stated in Section 3.2, the mean

pooling method averages the vectors of each token as a vector of the whole user review.

To be specific, as shown in Figure 3.2, firstly, my further pre-trained Game Specific Language Model will be used to generate token embeddings. After that, the mean pooling method provided by the Sentence Transformer [83] will be used to calculate the vector of the whole review by calculating the mean of all the token embeddings generated using the further pretrained Game Specific Language. The parameters used in vectorization are shown in Table 3.4.

Table 3.4: Text Vectorization Parameters

Parameters	**Details**
max_seq_length	500
word_embedding_model	Game Specific Language Model
Pooling Strategy	mean pooling

3.3.3 Implementation of clustering part

The main steps of the clustering part are:

1. Clustering the vectorized experimental instances using K-means.
2. Output the instances near the center of each cluster as a Seed Base Set.
3. Randomly select the instances as seed instances in the Seed Base Set.

Firstly, the sklearn.cluster.KMeans class from scikit-learn [87] [88] is used to cluster the vectorized user reviews. On top of K-Means, a method

for outputting the instances closest to each cluster's centroid is added. This method first transforms user reviews into a cluster-distance space using the Transform method in the KMeans class. Then it computes the top N vectorized instances of each cluster, and later outputs the natural language user reviews to a CSV file. The procedure of the method for outputting the instances closest to each cluster centroid is shown in Algorithm 1. Table 3.5 shows the parameters that were used in the clustering part. I used the elbow method [89] to decide the k-values for my experiment. Other k-values can be used based on classification criteria and data from prior knowledge or methods such as the elbow method. The purpose of this clustering step is not towards classification, but aiming at filtering noise and selecting a more representative dataset as the Seed Base Set, and does not require that all instances of the same category be in the same cluster, so the choice of k-value is relatively flexible.

Table 3.5: Clustering Parameters

Parameters	**Details**
Clustering Method	K means
Number of clusters(k)	5
Nearest points output threshold	60

The instances output by Algorithm 1 form the Seed Base Set, and the seed instances for forming the committee of the active learning part are randomly selected from the Seed Base Set.

3.3.4 Implementation of active learning part

This section details the implementation of the active learning design stated in Section 3.2.4. In order to conduct the learning cycle shown in Figure

Algorithm 1: Outputting the instances closest to each cluster centroid

```
Input: Vectorized user reviews
Output: Reviews closest to each cluster centroid
clustering_model := KMeansFromScikitlearn ;
corpus_embeddings := VectorizedUserReviews ;
km := clustering_model.fit(corpus_embeddings) ;
N := NearestPointsOutputThreshold ;
for i in range(number of clusters) do
    d := km.transform(corpus_embeddings)[:, i] ;
    index := indices of the N smallest values in d ;
    for j in index do
        NearestpointsOutput.append(corpus[j]) ;
    end
    CSVNPOutPut := pd.DataFrame(NearestpointsOutput) ;
    CSVNPOutPut.to_csv() ;
end
```

3.4, two main functions are built: the composing committee function and the main querying loop.

3.3.4.1 Composing committee function

The workflow for composing the committee is illustrated in Figure 3.9. As shown in the figure, the composing committee function contains two parts: a training set sampling method and classifiers. The main functionality of the training set sampling method is to select ten sets of instances in the training set to train ten classifiers. The Bootstrapping method is used for the sampling method, which means that the original sample is drawn ten times in a put-back manner. Each draw forms a new sample, and the operation is repeated to form ten new samples to train the classifiers. As for the classifier, the logistic regression classifier from scikit-

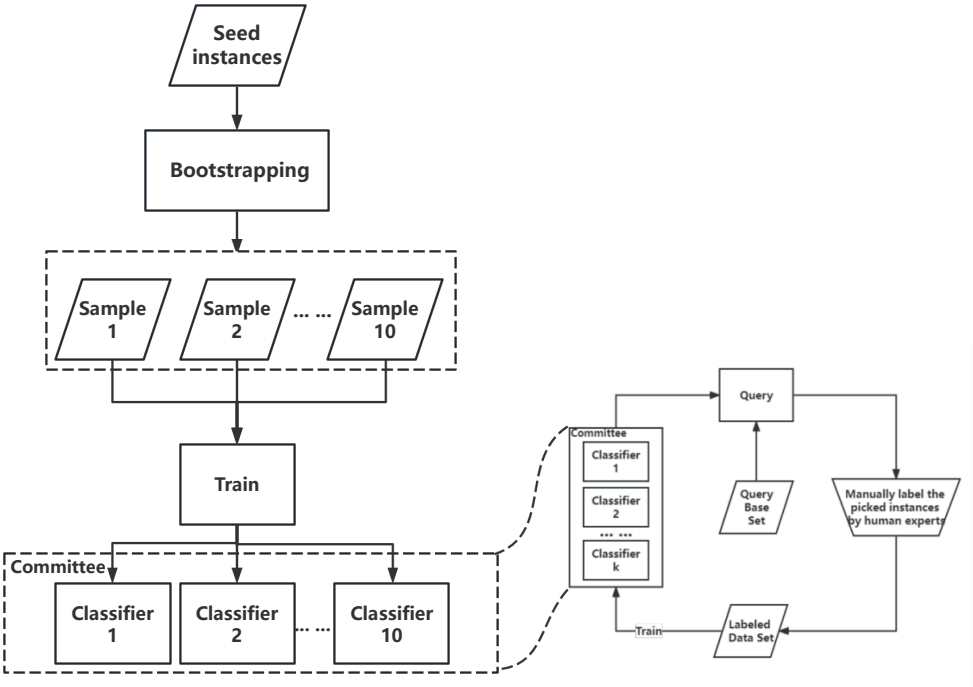

Figure 3.9: Workflow of composing committee

learn [87] [88] is used.

3.3.4.2 Main querying loop

The second main function of the active learning part is the main querying loop. The detailed procedure of the main querying loop is shown in Algorithm 2.

Firstly, the randomly selected ten seed instances from the Seed Base Set are passed into the composing committee function to form the committee.

Algorithm 2: Main querying loop

```
TrainingSet := SeedsSet ;
NumberofQueries := 0 ;
repeat
    Composing_committee(TrainingSet) ;
    Calculate_voting_entropy(QueryBaseSet) ;
    Selected_instances :=
     instances with the highest voting entropy ;
    Manually label the Selected_instances ;
    TrainingSet.update(Selected_instances) ;
    QueryBaseSet.difference_update(Selected_instances) ;
    logistic_regression.fit(TrainingSet) ;
    pred := logistic_regression.predict(TestSet) ;
    Calculate_accuracy_and_macro_F1(pred) ;
    NumberofQueries + + ;
until NumberofQueries = Specified number of queries;
```

As described in Section 3.2, the instances used to start active learning should be manually labeled. My Experimental Set was fully labeled during the data preparation phase, so the instances selected as seeds here will retain their labels to simulate the results of manual labeling.

Meanwhile, the instances in the Seed Base Set that are not randomly selected as seeds, along with the other experimental instances, are randomly split into two parts: 1500 instances (50% of all experimental instances) are assigned to the Test Set, and the remaining 1490 instances form the Query Base Set. Then the voting entropy introduced in Subsection 3.2.4.3 will be calculated. After that, the instances in the Query Base Set with the highest voting entropy will be selected for manual labeling. After labeling, the newly labeled instances will be put into the training set. The updated training set will be input to the composing committee function again to form an updated committee. Then the updated com-

mittee will be used to recalculate the voting entropy of the remaining instances in the Query Base Set to select the next instances to query.

According to the design, the instances that are included in the Query Base Set do not require labeling. Only when a specific instance is selected as a query and presented to human experts for inquiry, the instance needs to be labeled. Therefore, I temporarily remove the labels of the instances in the Query Base Set. When the active learning algorithm selects an instance as a query, I add its label back to simulate the process of manual labeling.

Also, in order to evaluate my system, during each query loop, the updated training set will be used to fit a logistic regression classifier, which will be used to give predictions to the instances in the Test Set and calculate the classification performance metrics.

3.4 Evaluation

This section describes how the evaluation was conducted and discusses the evaluation results.

3.4.1 Research questions

I pose the following two research questions for evaluating the proposed system:

- RQ1: How much does my system improve over the baseline approaches using the same amount of labeled instances?
- RQ2: Is the proposed system cost-efficient?

3.4.2 Evaluation for RQ1

3.4.2.1 Overview

For RQ1, the evaluation consists of two parts: the execution of the proposed system itself, and then the execution of other baseline approaches, which my system will be compared to.

3.4.2.2 RQ1 comparative experiments

This section describes the user review classification approaches I used to compare with the proposed system. Among the user reviews classification studies that have existed so far, the vast majority of methods consist of a text vectorization part and a classifier part [90].

In [90], Santos et al. analyzed the currently existing approaches for classifying software user reviews. Based on their study, I selected the two most commonly used text vectorization methods (TF-IDF, BoW) and the three most commonly used classifiers (SVM, Naive Bayes, Decision Tree) to form the software user review baseline approaches. In addition, I defined a set of PLM baseline approaches by fine-tuning BERT, ALBERT, and RoBERTa.

3.4.2.3 Evaluation Metrics

I use the accuracy score as the main evaluation metric to measure the performance of my system as well as the comparative approaches. The accuracy score is calculated by Formula (3.3), where *T11*, *T22*, *T33*, *F12*, *F13*, *F21*, *F23*, *F31*, *F32* are in the confusion matrix shown in Table 3.6. The highest value of the accuracy score is 1, and the lowest value is 0.

The larger the result, the better the classification performance.

$$\begin{aligned} \text{Accuracy} &= \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \\ &= \frac{T11+T22+T33}{T11+T22+T33+F12+F13+F21+F23+F31+F32} \end{aligned} \tag{3.3}$$

In addition, I use macro-F1 as a secondary metric to evaluate my system. The macro-F1 is calculated by Formula (3.4), where *T11*, *T22*, *T33*, *F12*, *F13*, *F21*, *F23*, *F31*, *F32* are also in the confusion matrix shown in Table 3.6. The highest value of the macro-F1 is 1, and the lowest value is 0. The larger the result, the better the classification performance.

$$\begin{aligned} \text{Macro-F1} &= \frac{\text{F1 Class }_1 + \text{F1 Class }_2 + \text{F1 Class }_3}{3} \\ &= \frac{\frac{2T11}{2T11+F21+F31+F12+F13}}{3} + \frac{\frac{2T22}{2T22+F21+F12+F32+F23}}{3} \\ &\quad + \frac{\frac{2T33}{2T33+F31+F32+F13+F23}}{3} \end{aligned} \tag{3.4}$$

Table 3.6: Confusion Matrix

Confusion Matrix		**Predicted Condition**		
		Class 1	Class 2	Class 3
Actual Condition	Class 1	T11	F12	F13
	Class 2	F21	T22	F23
	Class 3	F31	F32	T33

3.4.2.4 RQ1 experimental results

To ensure the confidence of the experimental results, I performed the experiment for the active learning part ten times to get an average classi-

Table 3.7: RQ1 Data Split

Data Set	**Number of Instances**	**Ratio**
Test Set	1500	50%
Seeds Set	10	0.3%
Query Base Set	1490	49.7%

fication performance. To be specific, each time, ten seed instances will be randomly selected from the Seed Base Set, and the split method will be called to form a Query Base Set and a Test Set. The specific data splits for RQ1 and their proportion of the total experimental instances are shown in Table 3.7.

As mentioned in Subsection 3.3.4.2, my system calculates classification performance at each query loop. Figure 3.10 illustrates the learning curve of my method, where the Y-axis represents the accuracy score and the X-axis denotes the number of iterations. Each iteration consists of five queries, meaning that in each loop, the five instances with the highest voting entropy are selected to form the batch. The red curve in the figure represents the learning curve of my system, illustrating the relationship between the number of queries and the resulting classification accuracy. The blue curve represents a control experiment that follows the same pipeline but employs random sampling instead. It can be observed that as the number of queries increases, the classification accuracy of my system steadily improves. Notably, my system achieves substantial gains in classification performance with only a small number of queries.

I selected the accuracy of 90 queries (18 query batches) after the learning curve flattened as the final result, which means that, together with the 10 seed instances, a total of 100 labeled instances were used for training. The performances of each fold and the average performance are shown

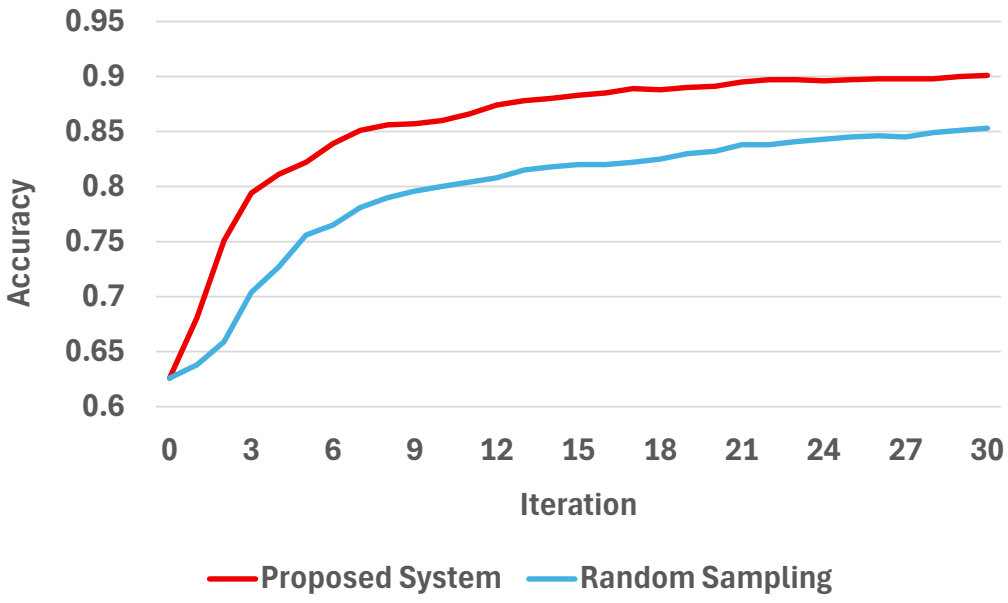

Figure 3.10: RQ1 learning curve

in Table 3.8. The results show that the proposed system using a total of only 100 labeled training instances can obtain an average accuracy score of 88.8%.

For comparison, I randomly selected the same number of labeled instances (100 instances) and test instances (1500 instances) in the Experimental Set to train and test the baseline approaches. This random sampling procedure was repeated ten times, and the average accuracy was calculated and reported in Table 3.9. It can be seen that among all the baseline models consisting of commonly used methods, the model with the highest accuracy achieves only 62.3% classification accuracy. In comparison, my system improves the classification accuracy by 26.5 percentage points, with a statistically significant difference ($p < 0.001$) compared to the best-performing baseline approach.

The experiment results show that my system, which combines further pre-training BERT, clustering, and active learning, is able to obtain a higher classification accuracy using a small number of labeled instances.

Table 3.8: RQ1 Performance

	Exp.1	**Exp.2**	**Exp.3**	**Exp.4**	**Exp.5**	**Exp.6**	**Exp.7**	**Exp.8**	**Exp.9**	**Exp.10**	**Average Performance ± Std.**
Acc	0.884	0.893	0.878	0.896	0.897	0.885	0.874	0.887	0.895	0.890	**0.888± 0.007**
F1	0.884	0.893	0.878	0.896	0.897	0.884	0.870	0.887	0.895	0.890	**0.887± 0.009**

Table 3.9: RQ1 Baseline approaches performance

Baseline Approaches	**Accuracy**
TF-IDF + SVM	0.542
TF-IDF + Naive Bayes	0.597
TF-IDF + Decision Tree	0.623
BoW + SVM	0.489
BoW + Naive Bayes	0.543
BoW + Decision Tree	0.592
Fine-tune BERT	0.571
Fine-tune RoBERTa	0.546
Fine-tune ALBERT	0.549

3.4.2.5 Answer to RQ1

According to the evaluation, my system can achieve 88.8% classification accuracy with 100 labeled training instances, while the highest accuracy among the baseline approaches is 62.3% using the same amount of labeled instances. The results show that my system successfully achieves the goal of achieving a higher classification accuracy with a small number of labeled instances.

3.4.3 Evaluation for RQ2

To further investigate the cost-efficiency of the proposed SE-SUA system, I conducted incremental experiments with the baseline approaches with labeled instance sizes ranging from 20 to 300, in incremental steps of 20, and compared the results with my system.

Specifically, I selected the best-performing software user review classification baseline approach from RQ1 (TF-IDF + Decision Tree) and the best-performing PLM baseline approach (fine-tuned BERT). For each step in the incremental experiments, randomly selected instances were added to the training sets of the baseline approaches, and I recorded the labeling cost, training time, and classification accuracy of both baseline approaches. To ensure fairness in the comparative experiments, the test set allocation ratio was kept consistent with that used in RQ1, and I also conducted ten independent incremental runs, reporting the average results. Regarding the computational environment, all experiments were conducted using the free tier of Google Colab, equipped with an NVIDIA Tesla T4 GPU and 15 GB of VRAM. For the calculation of labeling cost, in order to ensure fairness in the comparison experiments, I adopted the

crowdsourcing annotation price provided by Google Cloud for a single instance labeled by three annotators as the standardized unit of labeling cost. The results of the incremental experiments are visualized as cost-performance curves, as shown in Figure 3.11.

In the cost-performance curves plot, the results are visualized as two subplots for clarity. In the labeling cost vs. accuracy subplot, the x-axis represents classification accuracy, and the y-axis represents labeling cost (USD). In the training time vs. accuracy subplot, the x-axis also represents classification accuracy, while the y-axis corresponds to training time (in seconds). The blue lines correspond to the PLM baseline approach, the orange lines represent the software user review classification baseline approach, and the red lines indicate the results of the proposed system.

In the labeling cost vs. accuracy subplot, the curve of my system consistently lies below and to the right of both baseline approaches, without any intersection. This indicates that, at equivalent labeling costs, my system consistently outperforms both the PLM baseline and the commonly used software user review classification baseline in terms of classification accuracy, confirming the superior cost-efficiency of my system with respect to labeling effort.

In addition, the training time vs. accuracy subplot shows that throughout the incremental experiment, the training time of my system remains substantially lower than that of the PLM baseline. Since the difference in training time between my system and the commonly used software user review classification baseline is relatively small, I further calculated the average training time difference per step across the overlapping range of the incremental experiment. On average, my system takes only 0.63 seconds more per step than the traditional baseline, while achieving an

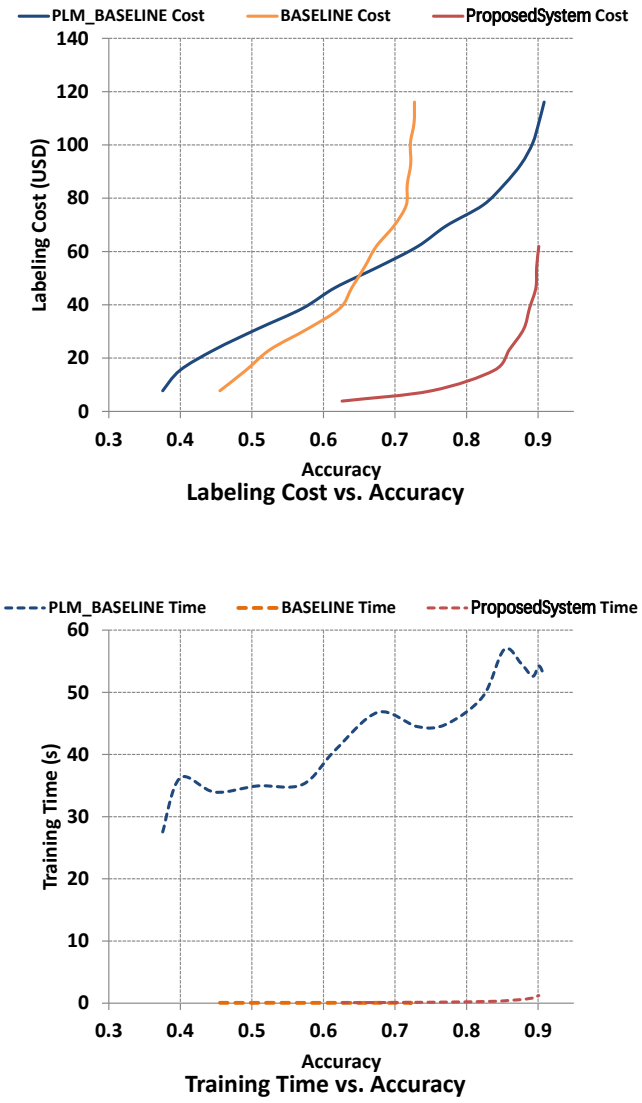

Figure 3.11: RQ2 cost-performance curves

average per-step training time reduction of 36.79 seconds compared to the PLM baseline. While the training time of my system is slightly higher than that of the traditional software user review classification baseline, the latter fails to achieve competitive accuracy throughout the entire range of the incremental experiment.

3.4.3.1 Answer to RQ2

As my system consistently demonstrates more efficient utilization of labeling cost along with relatively fast training speed, I conclude that the proposed SE-SUA system exhibits superior cost-effectiveness.

3.5 Threats to validity

3.5.1 Threats to construct validity

The threat to construct validity is mainly concerned with the data sets. The experimental instances used for evaluation are manually labeled. In order to mitigate this threat, two annotators labeled the instances. In the case where the two annotators differed in opinion, I invited a third annotator to vote for the correct label.

3.5.2 Threats to internal validity

The first threat to internal validity involves the classification factor I selected for classifying the user reviews, as the categories may not be completely independent of each other, with some overlap among them. In order to mitigate this threat, I infer an initial set of categories by ob-

serving user reviews, and then try to match these categories to different departments in game development, also referring to the classification factor in [86]. Furthermore, I assume that each user review belongs to only one of the categories I have defined. Reviews sometimes contain a primary intent and some secondary intents, so for each review, I tried to focus on the primary intent to allow developers to understand user needs and opinions more easily and prioritize their work accordingly. However, some information may be lost, thus posing a threat to the internal validity of my work.

Another threat to internal validity is the problem of overfitting in machine learning. To address this threat, I conducted the experiment ten times. I re-split the test set and training set randomly for each experiment and calculated the average performance.

3.5.3 Threats to external validity

The external threat to my work lies primarily in the ability to generalize my work. The training set used to train the Game Specific Language Model may not completely cover all game types. In order to mitigate this threat, I crawled 2,000 user reviews from each STEAM game category.

Finally, my choice was based on classifying the reviews based on which department in the game company should be checking the reviews. Different game companies may of course have different departments. Theoretically, my system will work on other classification factors, but future work should consider other classifications.

3.6 Chapter summary

In this chapter, I proposed a novel system for classifying game user reviews using only a small number of labeled instances. I evaluated the proposed system on user reviews collected from STEAM, and the results show that with just 100 labeled instances, my system achieved an average classification accuracy of 88.8%. In addition, I conducted a comprehensive cost-efficiency evaluation, which demonstrated the superior cost-effectiveness of my system. Moreover, to the best of my knowledge, this study is the first to explore the classification of game user reviews from a software engineering perspective, with the goal of supporting game developers. This is particularly important given that the online game sales market is already highly developed, with a vast number of users and user reviews available.

Although SE-SUA still relies on a small amount of manual annotation, it represents an important step toward the broader goal of this dissertation, achieving cost-effective text classification. At the same time, SE-SUA is not merely a transitional system designed to enable future lower-cost solutions; it also possesses unique advantages that make it suitable for application scenarios with stricter privacy constraints. First of all, by integrating self-supervised representations with clustering-guided seed selection and active learning, SE-SUA significantly reduces annotation costs compared to fully supervised pipelines. This demonstrates its potential as a foundation for advancing label efficiency in the zero-label setting explored in subsequent chapters. Through the combination of self-supervised representations, clustering-based seed selection, and developer-in-the-loop active learning, SE-SUA establishes a clear path to-

ward progressively eliminating manual supervision. Furthermore, SE-SUA operates without reliance on external APIs, which gives it a clearly defined system boundary and makes it well-suited for privacy-sensitive applications, such as internal feedback workflows.

Taken together, these characteristics position SE-SUA as a key contribution to the dissertation's agenda on cost-effective text classification.

Chapter 4

Cost-efficient LLM-driven multi-strategy active learning system for cross-task text classification

4.1 Introduction

Chapter 3 demonstrated that cost-efficient text classification can be achieved with minimal human supervision through the strategic combination of self-supervised, unsupervised, and active learning techniques. Building on the paradigm of reducing annotation costs to construct cost-efficient text classification systems, this chapter explores how to integrate large language models into the active learning framework in order to achieve text classification without relying on any manually labeled instances.

Recent advancements in LLMs, such as the Generative Pre-trained Transformer (GPT), have made it possible to perform zero-shot classification tasks via a question-and-answer paradigm. However, these models

were not originally designed for large-scale text classification, and they suffer from several limitations in this context [91,92]. As generative models, GPT-based systems require substantial computational resources for each classification task, since predictions are made sequentially through text generation [55]. Furthermore, inefficiencies in batch processing, invocation latency, and the limited throughput of current-generation GPT models exacerbate bottlenecks in large-scale classification scenarios, leading to higher costs and longer runtimes [61,62,93].

To address these limitations, I propose a cost-efficient, LLM-driven, multi-strategy active learning framework for cross-task text classification. The key idea is to use a generative LLM (e.g., GPT), guided by structured prompts, to automatically annotate instances selected by active learning query strategies—thus replacing the traditional human oracle. Additionally, I evaluate a range of query strategies and utilize RoBERTa to generate text embeddings that support both the query selection process and downstream classifier training.

This design enables my system to perform cross-task text classification—where "cross-task" refers to handling tasks with different objectives and label sets (e.g., sentiment analysis, topic categorization, and toxicity detection)—without relying on any human-annotated data. Furthermore, through a comprehensive comparison of different query strategies, classification tasks, and strategies for handling both the seed set and queried instances, I find that under my system design, the proposed LLM-driven active learning pipeline achieves performance competitive with traditional manually supervised active learning, while substantially reducing annotation costs. Importantly, it also avoids the inefficiencies and scalability issues associated with directly applying GPT to large-scale

classification tasks.

It is important to note that, active learning typically refers to a process in which a model selectively queries unlabeled data, obtains ground truth annotations from a human oracle, and iteratively updates itself using these annotations. In System 2, this definition is extended by treating pseudo labels generated by GPT as oracle-provided annotations for model updates. It should be emphasized that the "oracle" in this context does not denote a human expert supplying ground truth, but rather an automated pseudo-labeling mechanism.

4.2 Proposed system

I propose a cost-efficient LLM-driven multi-strategy active learning system for cross-task text classification without relying on human-annotated data. The overall workflow of my system is illustrated in Figure 4.1. Within the active learning loop, a query strategy selects informative instances from the unlabeled pool $\mathcal{U}$ and sends them to the GPT module for annotation. Guided by structured prompts, the GPT module labels the queried instances Q. The resulting labeled instances $\Delta\mathcal{L}$ are then added to the labeled set $\mathcal{L}$, which is used to update the machine learning classifier. In the following sections, I detail each component of this loop.

4.2.1 Query and update

The Query phase refers to the stage in the active learning loop where the system identifies the most informative instances from the unlabeled pool $\mathcal{U}$ to maximize learning efficiency. This is achieved through a query strat-

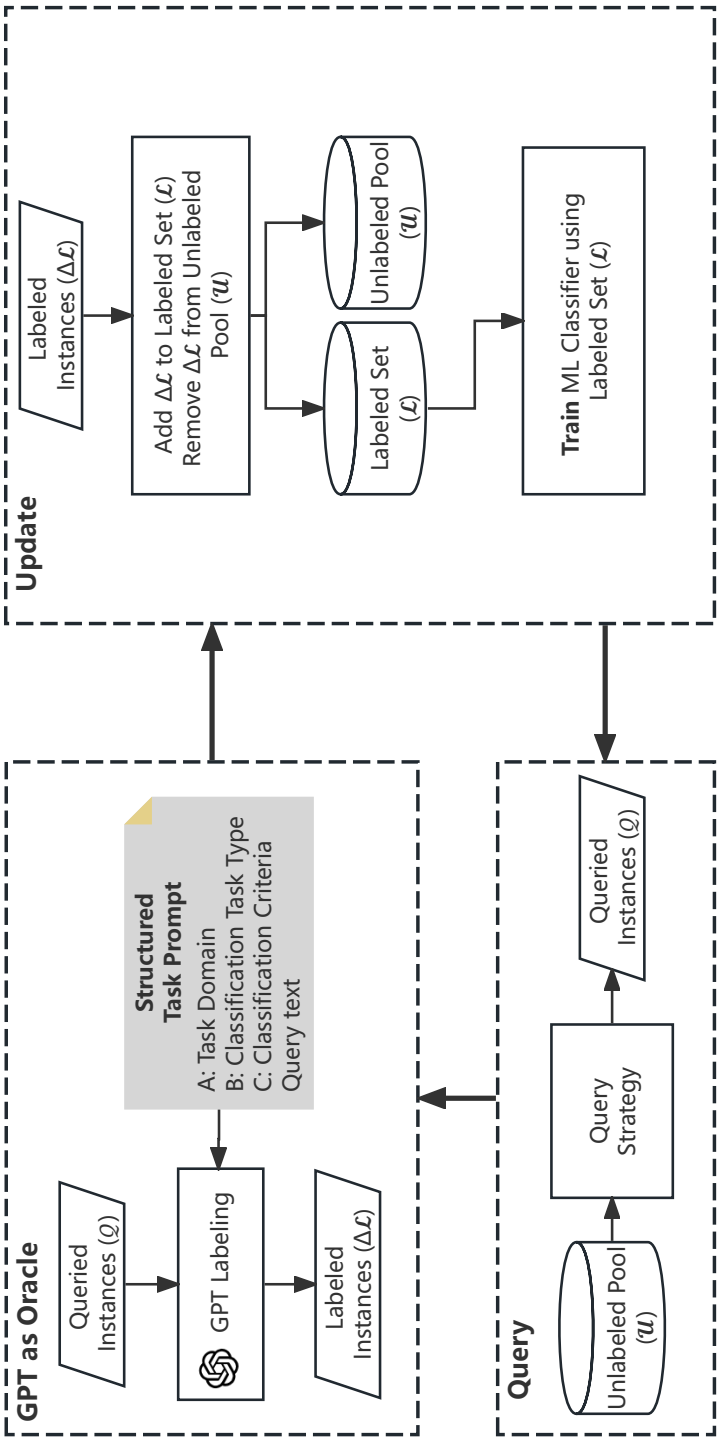

Figure 4.1: Overall workflow: LLM-driven active learning loop

egy that selects instances based on uncertainty, diversity, representativeness, or a hybrid of these criteria. The selected batch is then forwarded to the oracle for labeling.

In the Update phase, the newly labeled instances $\Delta\mathcal{L}$ are added to the labeled set $\mathcal{L}$ and removed from the unlabeled pool $\mathcal{U}$. The machine learning classifier is retrained on the updated labeled set. Before the first iteration, a seed set is randomly selected from the unlabeled pool $\mathcal{U}$ and labeled by the oracle to initialize the query strategy.

I implemented four widely studied query strategies: uncertainty sampling (Query by Committee) [43], entropy sampling with diversity [5,84], core-set sampling [45], and information density sampling [46]. Each strategy is evaluated independently in my experiments. I set the logistic regression classifier ($solver = lbfgs$, $multi\_class = auto$, $max\_iter = 1000$) as the default machine learning classifier, except for Query by Committee (QBC), which uses a voting classifier to form the committee. Additionally, to support query selection and classifier training, I utilize the roberta-base model with mean pooling to generate fixed-dimensional contextual embeddings. The subsequent subsections provide detailed descriptions of each query strategy.

4.2.1.1 Uncertainty sampling

For uncertainty sampling, I implemented a Query by Committee (QBC) approach using a soft voting classifier composed of four diverse models: a linear SVM (`Kernel=linear`, `Probability=True`), a decision tree (`Max Depth=None`), a random forest (`Estimators=100`, `Max Features=None`), and a logistic regression model (`solver=lbfgs`, `multi_class=auto`, `max_iter=1000`).

The committee models produce class probability distributions, which are averaged to reflect collective uncertainty.

To select the most informative samples for annotation, I adopt a batch-mode voting entropy criterion that quantifies the overall disagreement among committee members. Specifically, at each query round t, the algorithm aims to identify $Q_t^\star$, a subset of b unlabeled samples that maximizes the total voting entropy as shown in Formula (4.1):

$$Q_t^\star = \arg\max_{\substack{Q \subseteq U_t, \\ |Q|=b}} \sum_{x \in Q} \sum_{j=1}^{C} - \left(\frac{1}{M} \sum_{m=1}^{M} p_{m,j}(x) \right) \log \left(\frac{1}{M} \sum_{m=1}^{M} p_{m,j}(x) \right) \quad (4.1)$$

where, U_t is the current unlabeled pool, C is the number of classes, M is the number of committee members, and $p_{m,j}(x)$ denotes the predicted probability by model m that sample x belongs to class j. The inner summation computes the entropy of the committee-averaged class distribution for each sample x, which captures overall uncertainty. The batch $Q_t^\star$ thus contains the b samples with the highest cumulative voting entropy, representing the most uncertain instances under committee consensus.

4.2.1.2 Entropy sampling with diversity

For entropy sampling with diversity, I use the logistic regression model to estimate class probability distributions over the unlabeled pool. To select a batch of b samples that are both informative and diverse, the algorithm jointly considers two components: the predictive entropy of individual samples and the diversity among them. At each query round t, the algorithm aims to find the batch $Q_t^\star$ that maximizes the combination of uncertainty and diversity as defined in Formula (4.2):

$$Q_t^\star = \arg\max_{\substack{Q \subseteq U_t, \\ |Q|=b}} \sum_{x \in Q} \left[-\sum_{j=1}^{C} p_j(x) \log p_j(x) + \frac{1}{|Q|-1} \sum_{\substack{x' \in Q \\ x' \neq x}} d\left(\phi(x), \phi(x')\right) \right] \tag{4.2}$$

where, $p_j(x)$ denotes the predicted probability that sample x belongs to class j, C is the number of classes, and $\phi(x)$ represents the contextual embedding of sample x generated by the RoBERTa encoder. The first term in the summation computes the standard predictive entropy of sample x, reflecting uncertainty. The second term captures the average pairwise embedding distance between x and the other samples in the batch Q, promoting diversity. The samples with the highest combined scores are selected, resulting in a set that is both uncertain and diverse, thereby improving the model's generalizability by covering a broader region of the input space.

4.2.1.3 Core-set sampling

For core-set sampling, I leverage the pre-computed contextual embeddings from the RoBERTa encoder to represent each data point in a high-dimensional feature space. The aim is to select a batch of unlabeled samples that are most dissimilar from the currently labeled set, thereby covering new regions of the input space and improving the model's generalization. At each query round t, the algorithm selects the batch $Q_t^\star$ of b unlabeled samples that maximizes the minimum distance to the labeled set as defined in Formula (4.3):

$$Q_t^\star = \arg\max_{\substack{Q \subseteq U_t, \\ |Q|=b}} \sum_{x \in Q} \min_{l \in L_t} d\left(\phi(x), \phi(l)\right) \tag{4.3}$$

Here, U_t is the current unlabeled pool, L_t is the labeled set at round t, $\phi(x)$ denotes the embedding of sample x, and $d(\cdot, \cdot)$ is a distance metric (e.g., Euclidean distance). For each candidate x, the algorithm computes the distance to its nearest labeled sample and then sums these minimum distances over the batch. In conclusion, the samples with the largest minimum distance are chosen, allowing the model to expand its understanding by exploring less-represented areas of the data distribution.

4.2.1.4 Information density sampling

For information-density sampling, I aim to select a batch of unlabeled samples that are both uncertain and representative of densely populated regions in the feature space. This encourages the model to focus on samples that are not only ambiguous under current predictions but also structurally central within the data distribution. At each query round t, the algorithm selects the batch $Q_t^\star$ that maximizes the sum of entropy-to-density ratios as defined in Formula (4.4):

$$Q_t^\star = \arg\max_{\substack{Q \subseteq U_t, \\ |Q|=b}} \sum_{x \in Q} \frac{-\sum_{j=1}^{C} p_j(x) \log p_j(x)}{1 + \sum_{u \in U_t} \|\phi(x) - \phi(u)\|_2} \tag{4.4}$$

Here, $p_j(x)$ is the predicted probability that sample x belongs to class j, and $\phi(x)$ denotes the contextual embedding of sample x computed via RoBERTa. The numerator captures the predictive entropy of sample x, reflecting model uncertainty. The denominator represents the total distance from x to all other unlabeled samples, acting as the inverse of

information density. The final score for each sample is determined by multiplying the entropy score and the density score, ensuring that the selected samples are informative and representative of dense regions in the feature space.

4.2.2 GPT as oracle

GPT is used as an oracle in the active learning loop to provide labels for the queried instances. By leveraging GPT's extensive pre-trained knowledge, the need for human annotators is eliminated. To achieve this, prompt engineering is utilized to harness the question-answering capabilities of the GPT-4o model to issue labels. To ensure deterministic behavior and concise outputs suitable for labeling, I configured the GPT-4o model with *temperature* = 0 and *max_tokens* = 10 in all API calls. To ensure applicability across different tasks, I designed a structured prompt, as shown in Table 4.1. The prompt consists of several components: "A" specifies the classification task, "B" indicates whether it is a binary or multi-class classification, and "C" defines the classification standard. Finally, "query" contains the natural language text of instances selected by the active learning strategy. In conclusion, this design not only reduces reliance on human experts but also mitigates the limitations of using GPT directly for large-scale classification, such as high computation costs and long processing times.

Table 4.1: GPT Prompt Structure

Structured Prompt
{“role”: “system”, “content”: f“You are an expert in {A}.”}, {“role”: “user”, “content”: f“Now you have a {B}. Please {C}:‘{query}’. Please only return the label.”}

4.3 Evaluation

4.3.1 Research questions

To evaluate my system, I focus on the following research questions:

- RQ1: How does my system perform across different tasks without using any human-labeled samples?

- RQ2: What is the impact of using GPT as an oracle in active learning, compared to traditional human expert labeling?

- RQ3: What are the advantages of my system compared to directly using GPT for classification tasks?

- RQ4: What is the contribution of the active learning component in my system?

4.3.2 Data preparation

To evaluate my system, I prepared three widely used datasets for three different text classification tasks: the IMDB movie reviews dataset for sentiment classification, the AGnews dataset for news categorization, and the Jigsaw Toxic Comment Classification dataset for the toxic comment classification task.

Firstly, I preprocessed all three datasets to standardize the labels. In the IMDB dataset, “1” was used to represent positive sentiment, and ”0” for negative sentiment. In the AGnews dataset, labels were assigned as follows: “0” for World, “1” for Sports, “2” for Business, and “3” for Sci/Tech. For the Jigsaw Toxic Comment Classification dataset, “0” was used for non-toxic comments and “1” for toxic comments.

After label standardization, I randomly selected 10,000 instances from both the IMDB and AGnews datasets for my experiments. As for the Jigsaw Toxic Comment Classification dataset, since it had issues such as data with very few words and significant class imbalance, I further processed the dataset by removing extremely short instances—specifically, those with five words or fewer. After this filtering, I selected 5,000 instances each from toxic and non-toxic comments to create the final experimental dataset. The text length distributions of the three datasets are shown in Figure 4.2, Figure 4.3, and Figure 4.4. The label distributions are summarized in Figure 4.5.

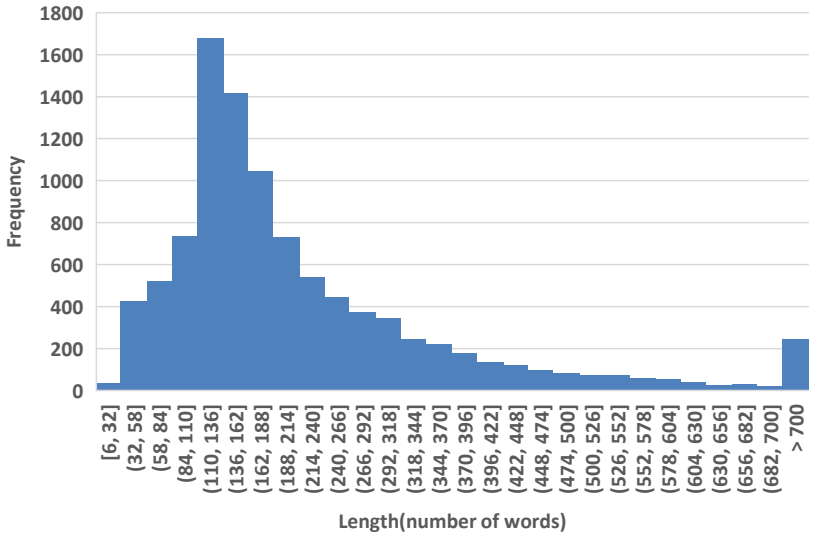

Figure 4.2: Length distribution of IMDB dataset

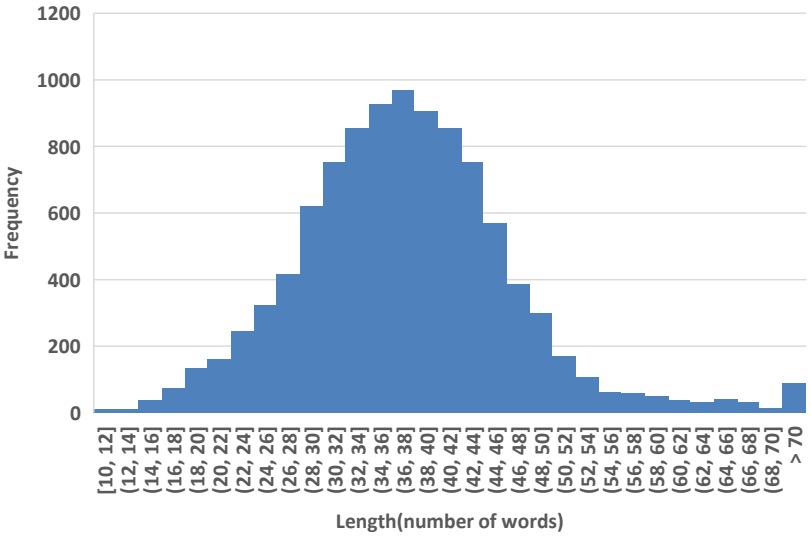

Figure 4.3: Length distribution of AGnews dataset

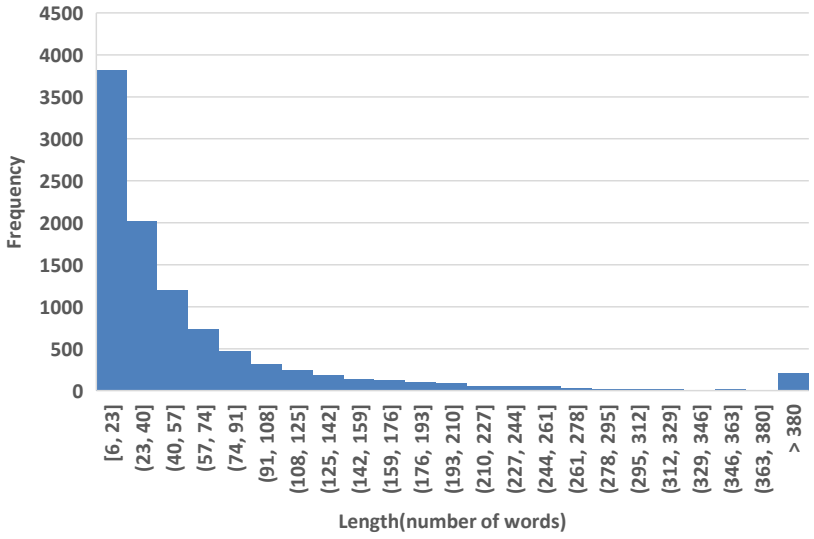

Figure 4.4: Length distribution of Jigsaw toxic comment classification dataset

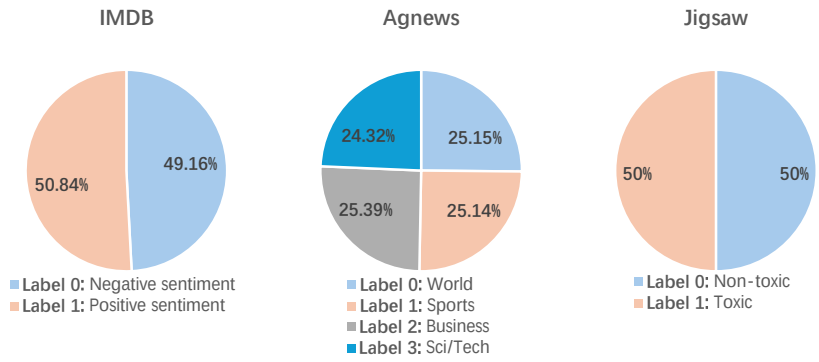

Figure 4.5: Label distributions

4.3.3 Experimental setup

The implementation details of my system have been discussed in Section 4.2. This section focuses on the evaluation metrics, experimental environ-

ment, and structured prompt settings for the experiments based on the research questions.

4.3.3.1 Evaluation metrics

I used accuracy as the primary evaluation metric, along with F1 score and recall to assess the performance of my experiments. The calculation of accuracy is given in Equation (4.5). The calculation of F1 score and recall is shown in Equation (4.6) and Equation (4.7).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{4.5}$$

$$\text{F1} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{4.6}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.7}$$

where TP (true positive) is the number of correctly predicted positive instances, FP (false positive) is the number of incorrectly predicted positive instances, and FN (false negative) is the number of missed positive instances.

4.3.3.2 Experimental environment

All experiments were conducted in the Google Colab environment, utilizing an NVIDIA A100-SXM4-40GB and an Intel(R) Xeon(R) CPU @ 2.20GHz, available to users on the lowest-tier plan of Google Colab.

4.3.3.3 Structured prompt settings

Based on the three datasets prepared for the classification tasks described in Section 4.3.2, I populated the structured prompt components A, B, and C, as shown in Table 4.1 of Section 4.2.2, for each task as follows:

- **IMDB Sentiment Classification Task:**
  - A = "user reviews sentiment classification"
  - B = "binary sentiment classification task"
  - C = "classify the following user review into positive sentiment or negative sentiment, use 1 for positive and 0 for negative"
- **AGnews News Classification Task:**
  - A = "news article classification"
  - B = "four-class news topic classification task"
  - C = "classify the following news article into one of the following categories: 0 for World, 1 for Sports, 2 for Business, or 3 for Sci/Tech"
- **Jigsaw Toxic Comment Classification Task:**
  - A = "toxic comment classification"
  - B = "binary classification task"
  - C = "classify the following comment into toxic or non-toxic, use 1 for toxic and 0 for non-toxic"

4.3.4 Evaluation for RQ1

RQ1: *How does my system perform across different tasks without using any human-labeled samples?*

To address RQ1, I used all three datasets and evaluated my system primarily using accuracy, with F1 score and recall as supplementary metrics. To ensure the robustness and reliability of the experimental results, I performed 5-fold cross-validation, with each fold using 80% of the dataset for training and 20% for testing.

In each fold of the cross-validation, I initially randomly selected 50 instances (representing 0.625% of the training set) from the training set as the seed set for starting active learning, and used structured prompts to request labels from GPT. I then began the active learning loop, setting the batch size to 5 and the number of iterations to 100. This means that each active learning loop consisted of 100 iterations, with each of the four query strategies independently selecting the top 5 most uncertain instances in each iteration, and requesting labels from GPT using my structured prompts. The obtained labels were then used to update each independent model.

Figures 4.6, 4.7, and 4.8 respectively show the learning curves of my system on the IMDB sentiment, AGnews, and Jigsaw toxic comment classification tasks. The x-axis represents the number of iterations, and the y-axis represents the primary evaluation metric—accuracy—which is averaged across 5-fold cross-validation. The red, green, purple, and blue lines represent the four query strategies combined with GPT. Additionally, in the IMDB learning curve (Figure 4.6), I include, as a control experiment, the fully automated variant of the pipelines proposed by Rouzegar

et al. [77]. It is worth noting that, unlike my method, Rouzegar et al.'s approach does not incorporate structures similar to my structured prompt design. As a result, their method relies on specifically designed prompts for each task, which limits its generalizability across tasks. Consequently, I was only able to evaluate their pipeline on the IMDB sentiment classification task, which is the only overlapping task with their study.

From Figure 4.6, I observed that among all evaluated strategies, the combination of GPT with entropy sampling with diversity achieved the best performance. Since active learning involves a trade-off between performance and labeling cost, I chose the 25th iteration—where performance gains begin to plateau—as the final evaluation point for the IMDB task, as recorded in Table 4.2. In contrast, the light blue curve corresponding to the Rouzegar et al. [77] approach shows consistently lower performance across the entire active learning process, underperforming all of my pipelines.

From Figure 4.7, I observed that for the AGnews news classification task, both information density and entropy sampling with diversity demonstrated similar strong performance. I chose the data from the 30th iteration, where the performance growth of both methods began to plateau, and selected the higher performance (with information density showing a slight advantage at that point) as the final performance of my system for the news classification task. The detailed results are recorded in Table 4.3.

From Figure 4.8, I observed that for the Jigsaw toxic comment classification task, entropy sampling with diversity demonstrated an advantage. Similarly, since active learning involves a trade-off between labeling cost and performance, I selected the result from the 19th iteration, where the

performance growth began to plateau, as the final toxic comment classification performance of my system. The detailed results are presented in Table 4.4.

In summary, my system achieved notable classification performance without any human-labeled data. Specifically, on the IMDB sentiment classification task, my method obtained an accuracy of 85.42% by using GPT for labels on 175 instances. For the AGnews news classification task, an accuracy of 84.88% was achieved by using 200 labeled instances from GPT. Lastly, for the Jigsaw toxic comment classification task, an accuracy of 86.44% was obtained by using 145 labeled instances from GPT. These results demonstrate that my system effectively achieves high classification performance across different tasks without relying on human annotations, proving both its effectiveness and its cross-task generalization capability.

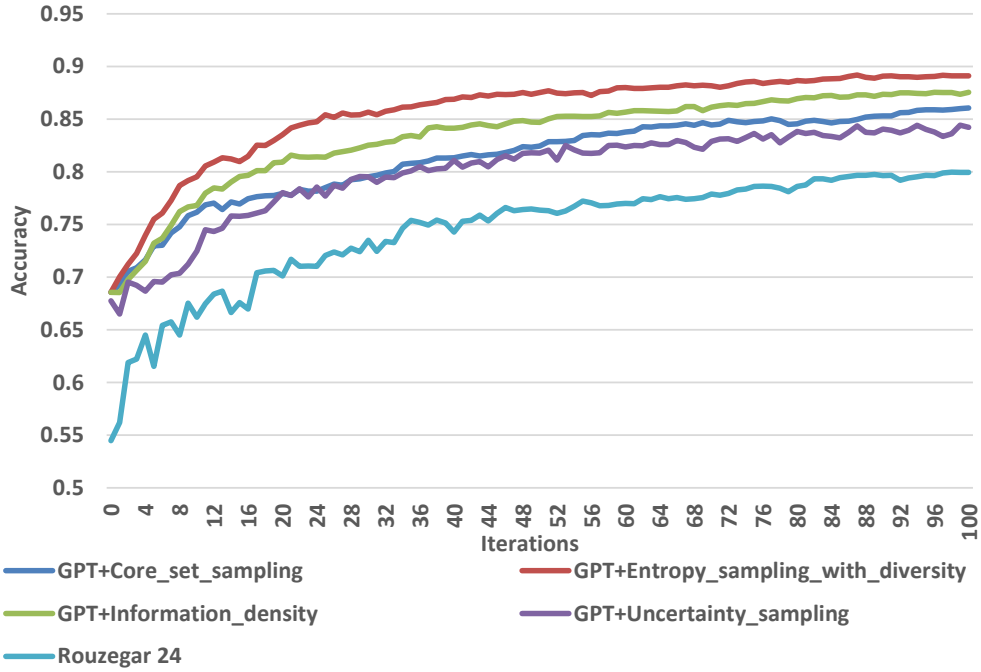

Figure 4.6: RQ1 learning curve: IMDB sentiment classification

Table 4.2: RQ1 Result: IMDB Sentiment Classification

Fold	Accuracy	F1	Recall
1	0.8555	0.8555	0.8557
2	0.8685	0.8680	0.8683
3	0.8335	0.8335	0.8345
4	0.8635	0.8635	0.8638
5	0.8500	0.8497	0.8509
Average	**0.8542**	**0.8540**	**0.8546**

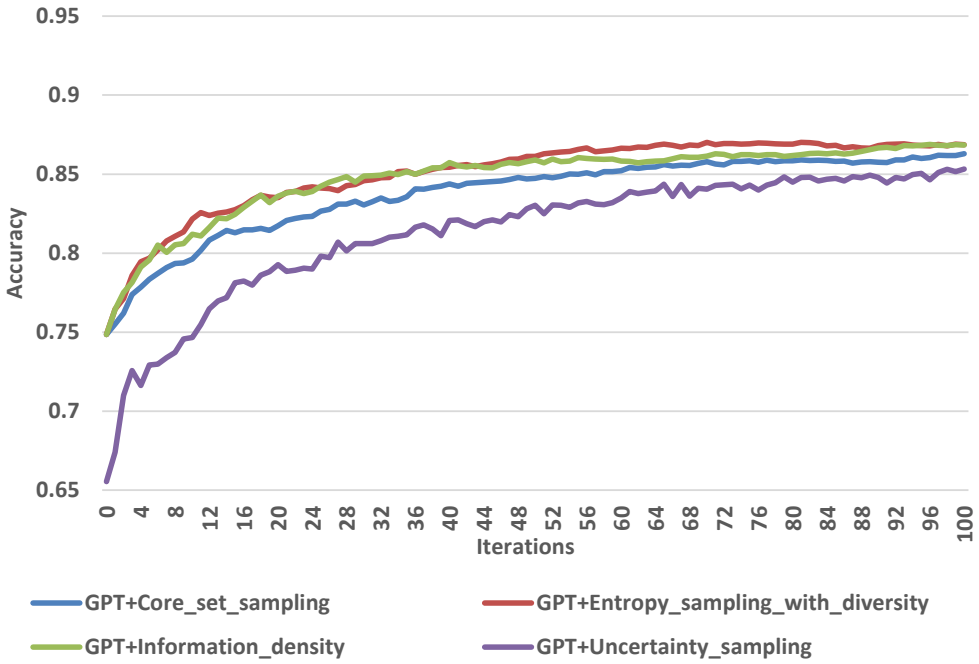

Figure 4.7: RQ1 learning curve: AGnews news classification

Table 4.3: RQ1 Result: AGnews news classification

Fold	Accuracy	F1	Recall
1	0.8345	0.8347	0.8351
2	0.8285	0.8252	0.8258
3	0.8500	0.8486	0.8495
4	0.8695	0.8680	0.8684
5	0.8615	0.8610	0.8604
Average	**0.8488**	**0.8475**	**0.8478**

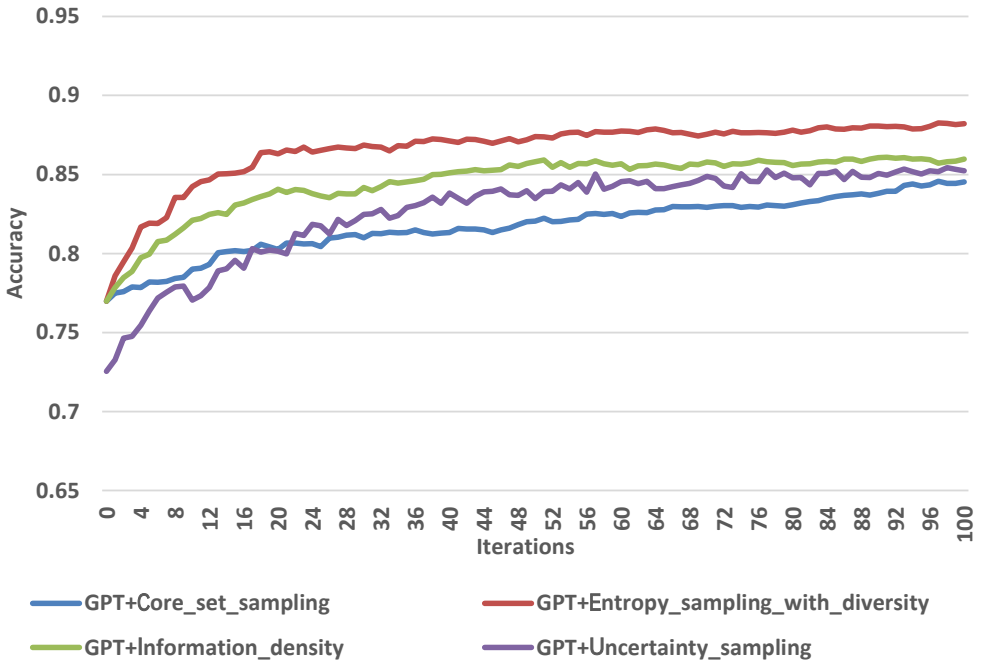

Figure 4.8: RQ1 learning curve: Jigsaw toxic comment classification

Table 4.4: RQ1 Result: Jigsaw Toxic Comment Classification

Fold	**Accuracy**	**F1**	**Recall**
1	0.8450	0.8442	0.8449
2	0.8665	0.8665	0.8666
3	0.8705	0.8704	0.8704
4	0.8725	0.8721	0.8721
5	0.8675	0.8674	0.8679
Average	**0.8644**	**0.8641**	**0.8644**

4.3.5 Evaluation for RQ2

RQ2: *What is the impact of using GPT as an oracle in active learning, compared to traditional human expert labeling?*

4.3.5.1 RQ2 setup

To evaluate the impact of GPT as an oracle in active learning, I compared the following three scenarios:

1. **Full GPT Labeling (my system)**: GPT is used for labeling both the seed set and all queries.
2. **Hybrid Labeling**: Human annotators provide labels for the seed set, while GPT labels all queries.
3. **Full Human Labeling**: Human annotators label both the seed set and all queries.

For scenarios requiring human-labeled samples, I directly used the labels from the original dataset. To ensure fairness in the comparative experiments, I used the same 5-fold splits and conducted experiments on the three classification tasks using the same experimental setup as in RQ1.

4.3.5.2 RQ2 results

Figures 4.9, 4.10, and 4.11 respectively show the comparative learning curves for the four query strategies on the IMDB sentiment, AGnews, and Jigsaw toxic comment classification tasks. Similar to RQ1, the y-axis

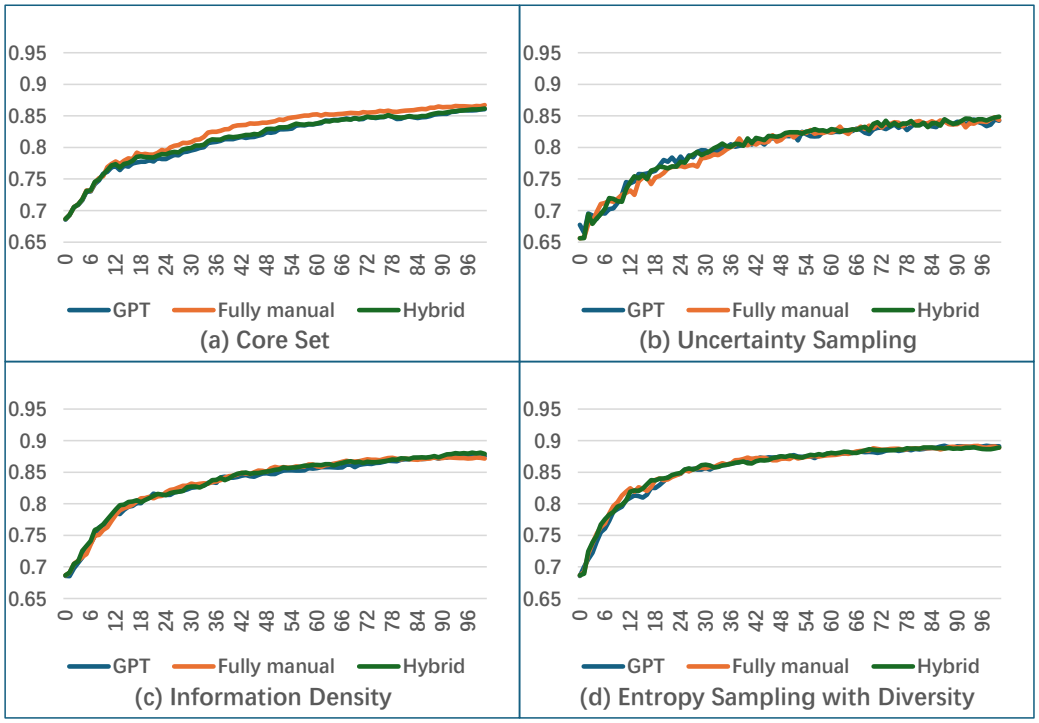

Figure 4.9: RQ2 learning curves: IMDB sentiment classification

represents the average accuracy from the 5-fold cross-validation, while the x-axis represents the number of iterations. Each figure contains four subplots, each representing a corresponding query strategy. In each subplot, three lines are shown, representing the cases of full GPT labeling, hybrid labeling, and full human labeling.

To provide a more detailed understanding of the comparative learning curves, I first calculated the standard deviation of the accuracy obtained from full GPT labeling, full human labeling, and hybrid labeling at each iteration. Then, I averaged these standard deviations across all iterations within each query strategy to obtain the average standard deviation for each query strategy in each of the three classification tasks. Detailed stan-

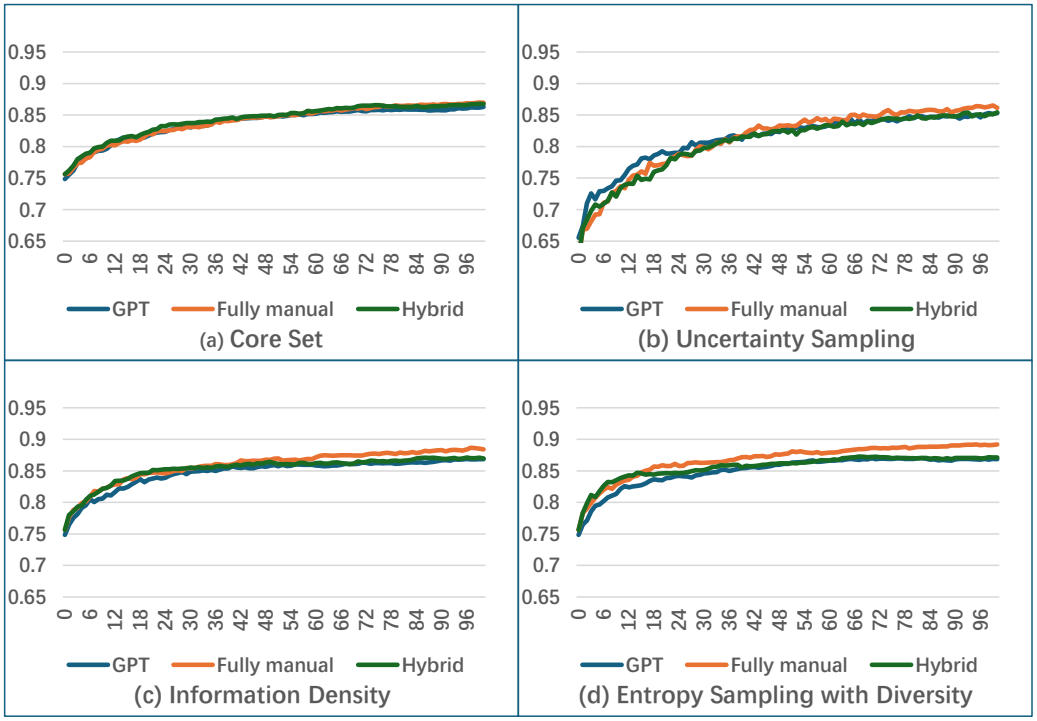

Figure 4.10: RQ2 learning curves: AGnews news classification

dard deviation results are listed in Table 4.5. For each query strategy on each task, the maximum cross iteration average standard deviation observed was 7.96×10^{-3}, and the overall average standard deviation across all tasks and query strategies was 4.06×10^{-3}. The relatively small average standard deviation across all query strategies and tasks indicates that the accuracy differences among full human labeling, hybrid labeling, and full GPT labeling remain stable across iterations.

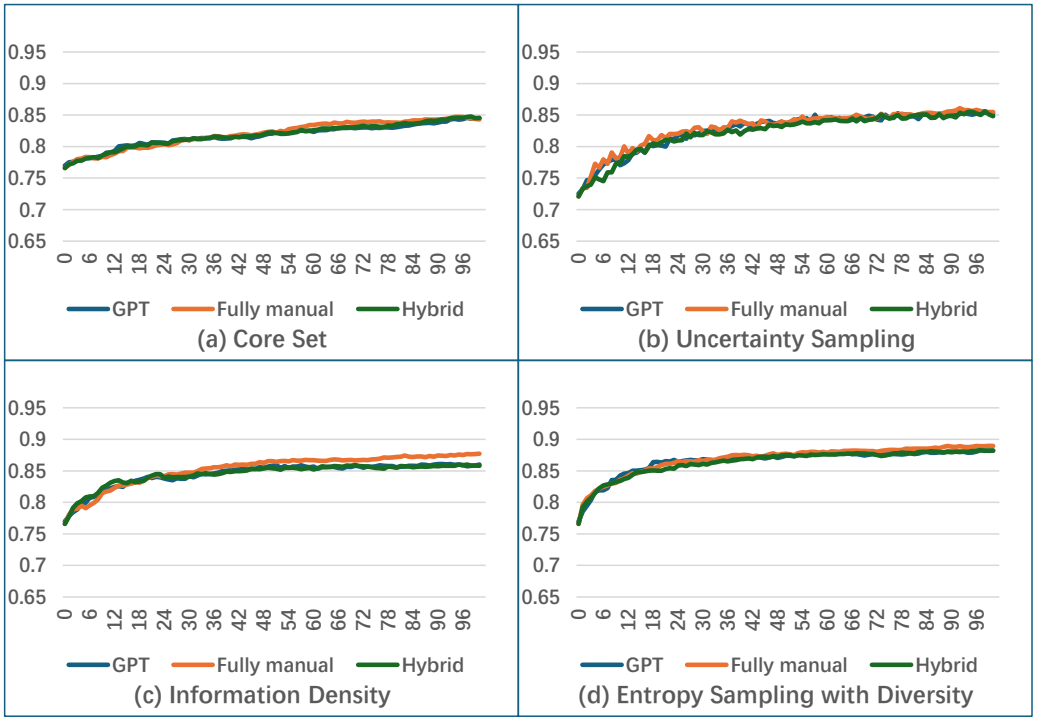

Figure 4.11: RQ2 learning curves: Jigsaw toxic comment classification

Table 4.5: RQ2 Result: Average Standard Deviation

	Core Set	**Entropy& Diversity**	**Information**	**Uncertainty**
IMDB	4.94×10^{-3}	2.14×10^{-3}	2.46×10^{-3}	3.87×10^{-3}
AGnews	2.55×10^{-3}	7.96×10^{-3}	5.31×10^{-3}	5.88×10^{-3}
Toxic	2.16×10^{-3}	2.82×10^{-3}	4.88×10^{-3}	3.72×10^{-3}

Furthermore, I calculated the accuracy differences between GPT labeling and full human labeling, as well as between GPT labeling and hybrid labeling, at each iteration. I then averaged the accuracy differences across iterations for each query strategy, yielding its average accuracy difference in each of the three classification tasks. Detailed accuracy difference

Table 4.6: RQ2 Result: Comparison of Average Accuracy Differences

		Core Set	**Entropy &Diversity**	**Information**	**Uncertainty**
GPT vs Fully Manual	IMDB	-0.011189	-0.002010	-0.002209	0.001413
	AGnews	-0.002059	-0.017321	-0.011646	0.000159
	Toxic	-0.002556	-0.003018	-0.007780	-0.003796
GPT vs Semi-manual	IMDB	-0.002429	-0.001682	-0.003287	-0.001831
	AGnews	-0.004733	-0.005985	-0.005905	0.006666
	Toxic	-0.000639	0.001758	-0.000205	0.003101

results are shown in Table 4.6, where negative values indicate that the accuracy of the active learning loop using full GPT labeling was lower, whereas positive values indicate higher accuracy. The worst case for full GPT labeling against full human labeling was -1.7321 percentage points, while the average across all tasks and strategies was -0.5168 percentage points. Similarly, the worst case of using full GPT labeling against hybrid labeling was -0.5985 percentage points, and the overall average was -0.1264 percentage points. These results indicate that under my system design, the proposed pipeline—GPT as Oracle, in which all seed and queried instances are labeled by GPT—achieves performance comparable to both fully manual and hybrid labeling pipelines.

To further analyze the cost-efficiency of the three labeling scenarios, I plotted the relationship between classification accuracy and labeling cost for the best-performing pipeline identified in RQ1 under each classification task. The resulting cost-performance curves are shown in Figure 4.12, which consists of three subplots corresponding to the IMDB sentiment, AGnews topic, and Jigsaw toxic comment classification tasks.

In each subplot, the x-axis represents classification accuracy, while the y-axis represents cumulative labeling cost. For instances labeled by humans, I adopted the unit cost defined by Google Cloud's crowdsourcing pricing, assuming three annotators per instance. For instances labeled by GPT, the labeling cost was computed based on API usage. Each subplot includes three lines: the blue line represents the fully automated GPT as Oracle active learning pipeline; the green and red lines represent the hybrid labeling and fully manual labeling pipelines, respectively.

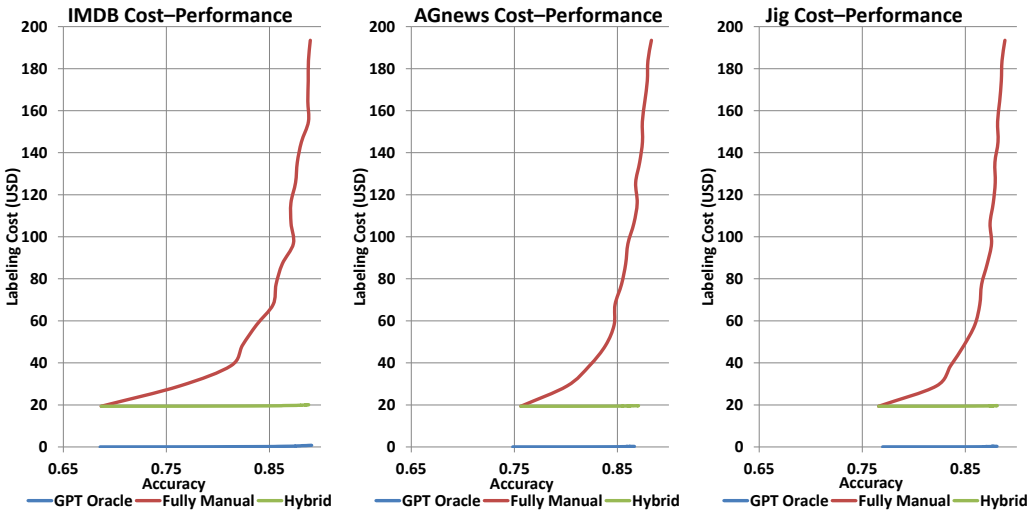

Figure 4.12: RQ2 cost-performance curves

As visualized in the cost-performance figure, the labeling cost of the full GPT labeling active learning pipeline remains substantially lower than that of both hybrid and fully manual labeling active learning throughout the entire process. To quantify this advantage, I computed the average proportion of labeling cost incurred by GPT-as-Oracle relative to the cost of hybrid and fully manual labeling across iterations. On the IMDB sentiment classification task, the full GPT labeling active learning incurred

only 1.92% of the cost of the hybrid labeling and 0.32% of the cost of full manual labeling active learning. For AGnews, the cost was 0.69% of hybrid and 0.12% of manual labeling. On the Jigsaw toxic comment classification task, the relative cost was 0.81% of hybrid and 0.14% of manual labeling. These results further demonstrate that fully automated, LLM-driven active learning can deliver competitive classification performance while drastically reducing annotation costs.

4.3.5.3 RQ2 conclusion

Based on these findings, I conclude that there is no substantial difference in the overall performance of active learning when using full human labeling, hybrid labeling, or fully GPT-labeled data. However, the full GPT labeling active learning pipeline offers a substantial advantage in terms of labeling cost, achieving comparable accuracy at only a fraction of the annotation expense. It is also important to note that in simulating human labeling, I relied on the ground truth annotations provided in the original datasets. Given that these datasets are well-established and have been widely used in academic research, I assume the human-labeled data to be of high quality. In real-world scenarios, however, human labeling may not consistently meet such rigorous standards, further highlighting the superior cost-efficiency of the full GPT labeling active learning pipeline.

4.3.6 Evaluation for RQ3

RQ3: *What are the advantages of my system compared to directly using GPT for classification tasks?*

To address RQ3, I evaluated the cost-efficiency of the proposed system

by comparing it with a baseline where GPT is directly used as a classifier without any active learning or model training. Specifically, I employed the structured prompt described in Section 4.3.3.3 to obtain predictions for the test set.

To ensure fairness in the comparative experiments, I used the same 5-fold cross-validation split as in RQ1. For each dataset, I recorded the following metrics:

- **Classification Accuracy**: The average accuracy across five folds.
- **Time Expenditure**: The average classification time across five folds.
- **Monetary Cost**: The average cost of GPT API usage across five folds, calculated based on the number of tokens processed.

Since direct GPT-based zero-shot classification does not involve any model training, the GPT classification time expenditure is the inference time. In contrast, for my system, the total time expenditure includes both model training time and inference time, to reflect the full computational effort involved. Figure 4.13 and Table 4.7 summarize the 5-fold average results across the three datasets in terms of classification accuracy, time expenditure, and monetary cost. In each subplot of the figure, blue bars represent the direct GPT classification baseline, while red bars correspond to the proposed LLM-driven active learning system.

Based on the experimental results, my system required only approximately 6% of the time and monetary cost compared to directly using GPT for classification, while still achieving over 93% of its classification performance. These findings demonstrate that my method offers a highly

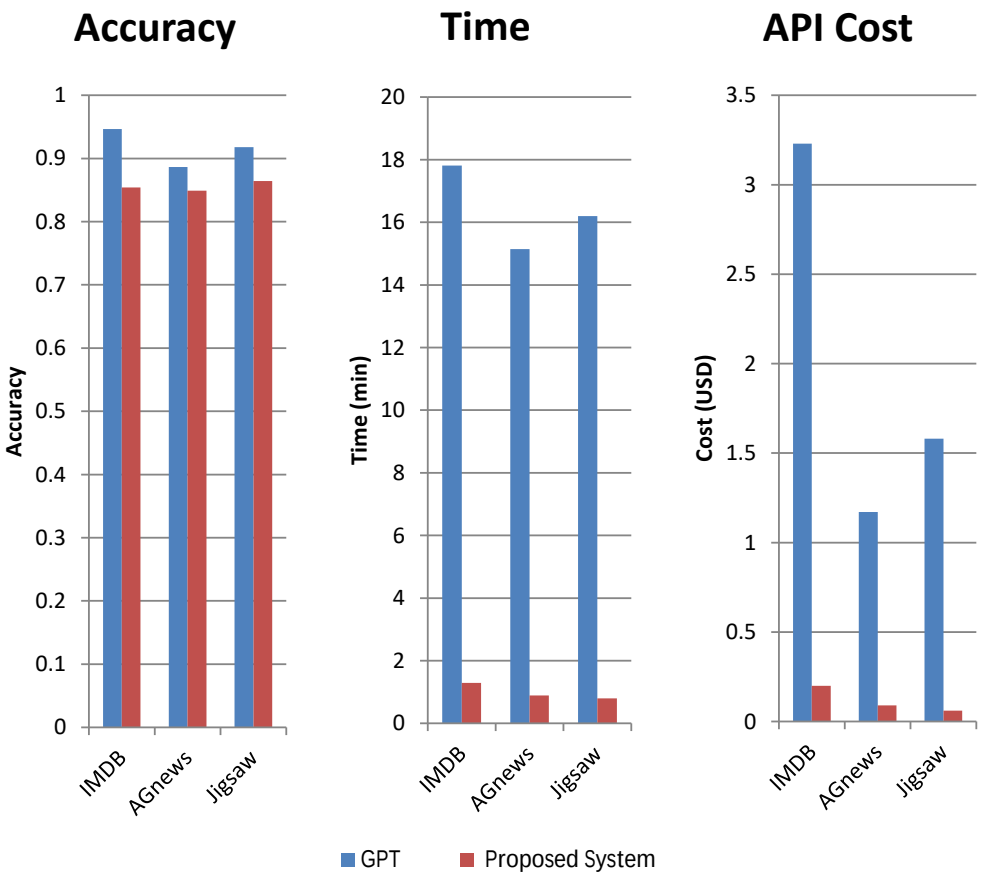

Figure 4.13: RQ3 cost-efficiency comparison

cost-effective alternative to direct GPT classification, making it particularly suitable for deployment in cost-sensitive or resource-constrained environments.

4.3.7 Evaluation for RQ4

RQ4: *What is the contribution of the active learning component in my system?*

Table 4.7: RQ3 Result: Comparison Between My System and Direct GPT Usage

	Direct GPT Usage			**My System**		
	Accuracy	Time (min)	Cost (USD)	Accuracy	Time (min)	Cost (USD)
IMDB	0.9463	17.81	3.23	0.8542	1.29	0.20
AGnews	0.8866	15.14	1.17	0.8488	0.89	0.09
Jigsaw	0.9178	16.20	1.58	0.8644	0.80	0.06

To answer RQ4, I disabled the query strategy components and instead randomly selected the same number of instances from the training set as the combined total of the seed and query instances used in each of the final performance settings of RQ1. I used the sampled instances to train the default logistic regression classifier from the active learning module. To ensure the rigor of the comparative experiment, I conducted the same 5-fold cross-validation, and for each fold, I repeated the random sampling five times, taking the average as the result for that fold. The detailed classification accuracy results are shown in Table 4.8.

Table 4.8: RQ4 Result: Classification Accuracy With and Without Active Learning

Task	**Without AL**	**With AL**	**Accuracy Gain (pp)**
IMDB	77.56%	85.42%	+7.86
AGnews	80.29%	84.88%	+4.59
Toxic	80.02%	86.44%	+6.42

As observed, incorporating the active learning component led to an absolute average classification accuracy gain of 7.86 percentage points ($p < 0.01$) in the IMDB sentiment classification task, 4.59 percentage

points ($p < 0.001$) in the AGnews news classification task, and 6.42 percentage points ($p < 0.001$) in the Jigsaw toxic comment classification task. This demonstrates that active learning contributes to model performance by guiding the selection of informative instances. Therefore, I conclude that integrating active learning into my system is beneficial, as it enhances classification accuracy with a limited number of labeled instances.

4.4 Chapter summary

This study proposes a cost-efficient text classification system that integrates large language models (LLMs) with active learning, enabling effective cross-task text classification without requiring manually labeled data. Experimental results confirm that my system successfully eliminates the need for human annotations while ensuring high classification performance across diverse tasks.

Furthermore, I systematically evaluated the performance of different query strategies within the LLM-driven active learning framework. At the same time, I conducted a comprehensive analysis of different labeling scenarios for seed and queried instances, including full GPT labeling, hybrid labeling, and traditional fully manual active learning. The results show that the proposed automated active learning pipeline using full GPT labeling achieves performance comparable to traditional fully manual active learning, while exhibiting superior cost-efficiency. Moreover, my experiments demonstrate that, compared to directly using generative AI models like GPT for text classification, my system achieves 93% of its classification performance while requiring only about 6% of the time and monetary cost.

In terms of practical implications, my system can be particularly valuable in resource-constrained scenarios where labeled data is scarce or manual annotation is infeasible. To evaluate its generalizability, I conducted experiments on three distinct classification tasks: sentiment analysis, topic categorization, and toxicity detection. These tasks differ in their semantic focus and label spaces, providing initial evidence of the system's robustness across task types. This demonstrated cross-task generalization capability broadens the potential applicability of my system to a wide range of text classification scenarios.

Chapter 5

Conclusion

5.1 Overall summary

This dissertation, from the perspective of software engineering for machine learning, identifies the cost-efficiency challenges of machine learning–based text classification systems. To address these issues, I introduced two complementary, cost-efficiency-oriented solutions that together demonstrate how strategic combinations of self-supervised learning, unsupervised learning, active learning, and large language models (LLMs) can deliver competitive performance while achieving superior cost-effectiveness.

Chapter 3 presented the SE-SUA system, a cost-efficient classification framework for software-engineering-oriented game user reviews. By integrating self-supervised representation learning, unsupervised clustering, and active learning, SE-SUA significantly outperforms baseline approaches under the same number of labeled instances. A comprehensive cost-efficiency analysis further confirms that the SE-SUA system achieves high classification performance while substantially reducing annotation costs and maintaining relatively fast training speed, demonstrating its su-

perior cost-efficiency. Notably, SE-SUA not only validates the paradigm of minimal supervision as a foundation for future fully automated systems, but also offers practical advantages in real-world applications: its design does not rely on external APIs, which establishes clear system boundaries and makes it particularly suitable for deployment in privacy-sensitive scenarios. These characteristics highlight SE-SUA's potential as a standalone, complementary solution for text classification under constrained resources.

Chapter 4 proposed a cost-efficient, LLM-driven, multi-strategy active learning system for cross-task text classification that requires no human-labeled data. The system uses structured prompts to guide GPT as the oracle within the active learning loop, and incorporates four distinct query strategies to select the most informative instances. Comprehensive experiments demonstrate that, under my system design, the proposed fully GPT-labeled active learning pipeline achieves performance comparable to traditional fully manual active learning, while significantly reducing annotation costs. Furthermore, compared to directly using GPT for classification, the proposed system achieves approximately 93% of its classification performance while incurring only about 6% of the associated time and monetary cost. Beyond proposing a cost-efficient classification system that eliminates the need for human annotation, Chapter 4 conducts comprehensive ablation studies to evaluate the performance of different query strategies and alternative treatments of seed and queried instances under the LLM-driven active learning setting. These findings further pave the way toward lowering the entry barrier for text classification, contributing to the broader goal of democratizing access to text classification technology.

In summary, this dissertation advances the state of cost-efficient text classification and offers actionable insights for practitioners aiming to deploy such systems in resource-constrained scenarios. By proposing two complementary systems, it demonstrates how principled software engineering for machine learning can guide the design of cost-efficient text classification pipelines, thereby lowering the barrier to adoption and paving the way for broader technological democratization.

5.2 Future work

Building on the contributions of this dissertation, several promising directions remain for future research and practical extension:

5.2.1 Multilingual extension

While this dissertation focused primarily on English-language datasets, the proposed systems are not inherently language-bound. A natural next step is to extend both systems—SE-SUA and the LLM-driven active learning pipeline—to multilingual and non-English contexts. This would involve adapting the language models, prompts, and pretraining corpora to support cross-lingual classification and evaluating performance across diverse linguistic settings. Supporting multilingual data would greatly enhance the global applicability of cost-efficient text classification systems.

5.2.2 Model efficiency and sustainability

Building upon the cost-efficient system frameworks established and validated in this dissertation, future work may further explore the use of more lightweight LLMs and seek a further balance between performance and cost. In addition, future research could incorporate environmental considerations, such as carbon footprint, into both evaluation and optimization, with the aim of developing systems that are not only cost-efficient but also environmentally sustainable.

5.2.3 Technology democratization

A long-term goal of this research is to contribute to the broader democratization of machine learning technologies. In future work, the proposed frameworks could be applied in real-world industrial or social contexts where labeled data is limited and resource constraints are prevalent. Deploying these systems in operational pipelines would allow for iterative refinement, uncover practical challenges in model integration, and provide valuable feedback for future design. Ultimately, this line of research seeks to make advanced text classification systems more accessible and impactful for a wider range of users and applications.

Acknowledgments

First of all, I would like to express my deepest gratitude to my lab advisor, Prof. Shingo Takada, for his invaluable guidance and support throughout the five years of my academic journey in the Takada Laboratory, from the master's to the doctoral program. Prof. Takada has provided me with comprehensive and meticulous academic supervision, and more importantly, has had a profound and positive influence on my ability to face challenges and setbacks and to think thoroughly and critically. I believe these influences will continue to benefit me in my future research and development career.

I would also like to extend my sincere thanks to my co-advisors, Prof. Komei Sugiura, Prof. Hiroaki Saito, and Prof. Takahiro Yakoh. Their thoughtful advice and continued support have greatly enhanced the quality and depth of my research. I am especially grateful for their patient guidance and ongoing help. What they have imparted to me goes far beyond improvements to this dissertation; it has deepened and refined my understanding of the research process as a whole and strengthened my confidence to tackle future challenges.

I would also like to thank the professors and administrative staff of the Faculty of Science and Technology at Keio University. The high-quality

courses, generous academic support, and exceptional care they provided, both academically and in daily life, made my experience as an international student incredibly fruitful and rewarding.

I would also like to express my sincere appreciation to the Fujiwara Scholarship Fund for their generous financial support. The scholarship has greatly helped me maintain stability in my daily life and has allowed me to focus more deeply on my academic pursuits.

Finally, I would like to express my heartfelt appreciation to my parents, friends, and labmates. Without their constant encouragement and unwavering support, this doctoral journey would not have been possible.

References

[1] F. Sebastiani, "Machine learning in automated text categorization," *ACM computing surveys (CSUR)*, vol. 34, no. 1, pp. 1–47, 2002.

[2] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, "A survey on text classification algorithms: From text to predictions," *Information*, vol. 13, no. 2, 2022.

[3] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: Sentiment classification using machine learning techniques," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.

[4] I. Androutsopoulos, J. Koutsias, V. Chandrinos, and C. D. Spyropoulos, "An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 160–167, ACM, 2000.

[5] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th annual international ACM SIGIR*

conference on Research and development in information retrieval, pp. 3–12, Springer-Verlag New York, Inc., 1994.

[6] C. Iacob and R. Harrison, "Retrieving and analyzing mobile apps feature requests from online reviews," in *Proceedings of the 10th Working Conference on Mining Software Repositories (MSR)*, pp. 41–44, IEEE Computer Society, 2013.

[7] S. Panichella, A. D. Sorbo, E. Guzman, C. A. Visaggio, G. Canfora, and H. C. Gall, "How can I improve my app? Classifying user reviews for software maintenance and evolution," in *Proceedings of the 31st IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 281–290, IEEE Computer Society, 2015.

[8] V. M. A. de Lima, J. R. Barbosa, and R. M. Marcacini, "Issue detection and prioritization based on mobile application reviews," *Software Quality Journal*, vol. 33, no. 1, pp. 1–35, 2024. First online 23 December 2024.

[9] D. Pagano and W. Maalej, "User feedback in the appstore: An empirical study," in *2013 21st IEEE international requirements engineering conference (RE)*, pp. 125–134, IEEE, 2013.

[10] F. Palomba, M. Linares-Vásquez, G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps," in *2015 IEEE international conference on software maintenance and evolution (ICSME)*, pp. 291–300, IEEE, 2015.

[11] J. A. Chevalier and D. Mayzlin, "The effect of word of mouth on sales: Online book reviews," *Journal of Marketing Research*, vol. 43, no. 3, pp. 345–354, 2006.

[12] Y. Liu, "Word of mouth for movies: Its dynamics and impact on box office revenue," *Journal of Marketing*, vol. 70, no. 3, pp. 74–89, 2006.

[13] N. Hu, P. A. Pavlou, and J. Zhang, "Overcoming the J-shaped distribution of product reviews," *Communications of the ACM*, vol. 52, no. 10, pp. 144–147, 2009.

[14] C. Dellarocas, "The digitization of word of mouth: Promise and challenges of online feedback mechanisms," *Management Science*, vol. 49, no. 10, pp. 1407–1424, 2003.

[15] C. Forman, A. Ghose, and B. Wiesenfeld, "Examining the relationship between reviews and sales: The role of reviewer identity disclosure in electronic markets," *Information Systems Research*, vol. 19, no. 3, pp. 291–313, 2008.

[16] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang, "Topic detection and tracking pilot study: Final report," in *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pp. 194–218, 1998.

[17] O. D. Clercq, L. de Bruyne, and V. Hoste, "News topic classification as a first step towards diverse news recommendation," *Computational Linguistics in the Netherlands Journal*, vol. 10, pp. 37–55, 2020.

[18] M. Abdullah and M. G. H. I. A. Zamil, "The effectiveness of classification on information retrieval system (case study)," *arXiv preprint arXiv:1804.00566*, 2018.

[19] E. Wulczyn, N. Thain, and L. Dixon, "Ex machina: Personal attacks seen at scale," in *Proceedings of the 26th International World Wide Web Conference (WWW)*, pp. 1391–1399, ACM, 2017.

[20] S. Jhaver, I. Birman, E. Gilbert, and A. Bruckman, "Human–machine collaboration for content regulation: The case of reddit's *automoderator*," *ACM Transactions on Computer–Human Interaction*, vol. 26, no. 5, pp. 31:1–31:35, 2019.

[21] P. Fortuna and S. Nunes, "A survey on automatic detection of hate speech in text," *ACM Computing Surveys*, vol. 51, no. 4, pp. 85:1–85:30, 2018.

[22] B. Vidgen, A. Harris, D. Nguyen, R. Tromble, S. Hale, and H. Margetts, "Challenges and frontiers in abusive content detection," in *Proceedings of the Third Workshop on Abusive Language Online*, (Florence, Italy), pp. 80–93, Association for Computational Linguistics, Aug. 2019.

[23] A. Schmidt and M. Wiegand, "A survey on hate speech detection using natural language processing," in *Proceedings of the Fifth International Workshop on Natural Language Processing for Social Media*, (Valencia, Spain), pp. 1–10, Association for Computational Linguistics, Apr. 2017.

[24] "Regulation (EU) 2022/2065 of the European Parliament and of the Council of 19 October 2022 on a Single Market for Digital Services and Amending Directive 2000/31/EC (Digital Services Act)." Official Journal of the European Union **L 277** (27 Oct 2022), 1–102, 2022.

[25] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[26] E. K. Ringger, M. D. Wood, B. D. Price, D. T. McClanahan, D. Lonsdale, T. Haertel, K. Seppi, and K. Pohs, "Assessing the costs of machine-assisted corpus annotation through a user study," in *Proceedings of the 6th International Language Resources and Evaluation Conference (LREC)*, 2008.

[27] C. G. Northcutt, L. Jiang, and I. Chuang, "Confident learning: Estimating uncertainty in dataset labels," *Journal of Artificial Intelligence Research*, vol. 70, pp. 1373–1411, 2021.

[28] B. Liu, X. Ren, *et al.*, "Learning with noisy labels for text classification: A comprehensive study," in *Proceedings of the 29th International Conference on Computational Linguistics (COLING)*, 2022.

[29] Q. Wei, Z. Ji, T. Li, and Z. He, "Clinical text annotation—what factors are associated with the cost of time?," in *AMIA Annual Symposium Proceedings*, pp. 1136–1145, 2018.

[30] H. Chau, M. Pham, M. Nguyen, and N. Luong, "Understanding the trade-off between cost and quality of expert annotations for keyphrase extraction," in *Proceedings of the 14th Linguistic Annotation Workshop*, pp. 134–139, 2020.

[31] S. Amershi, A. Begel, C. Bird, R. DeLine, H. Gall, E. Kamar, N. Nagappan, B. Nushi, and T. Zimmermann, "Software engineering for machine learning: A case study," in *Proceedings of the 41st Interna-*

tional Conference on Software Engineering: Software Engineering in Practice, pp. 291–300, 2019.

[32] L. Liao, T. Chen, B. Zhang, B. Hong, E. Shihab, and Y. Zou, "An empirical study of the impact of hyper-parameter tuning on deep learning performance," *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 4, pp. 42:1–42:31, 2021.

[33] A. Arpteg, B. Brinne, L. Crnkovic-Friis, and J. Bosch, "Software engineering challenges of deep learning," in *Proceedings of the 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 50–59, 2018.

[34] J. H. Bernardo, D. A. da Costa, F. R. Cogo, S. Q. de Medeiros, and U. Kulesza, "Continuous integration practices in machine learning projects: The practitioners' perspective," *arXiv preprint arXiv:2502.17378*, 2025.

[35] K. Shivashankar, G. Al Hajj, and A. Martini, "Maintainability and scalability in machine learning: Challenges and solutions," *ACM Comput. Surv.*, vol. 57, July 2025.

[36] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, and D. Dennison, "Hidden technical debt in machine-learning systems," in *Advances in Neural Information Processing Systems 28*, pp. 2503–2511, 2015.

[37] C. Rudin, "Stop explaining black box machine learning models for high-stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[38] S. R. Hong, J. Hullman, and E. Bertini, "Human factors in model interpretability: Industry practices, challenges, and needs," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW1, pp. 32:1–32:26, 2020.

[39] D. Angluin, "Queries and concept learning," *Machine learning*, vol. 2, pp. 319–342, 1988.

[40] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Machine learning*, vol. 15, pp. 201–221, 1994.

[41] N. Abe, "Query learning strategies using boosting and bagging," *Proceedings of the International Conference on Machine Learning 98*, pp. 1–9, 1998.

[42] D. Tuia, F. Ratle, F. Pacifici, M. F. Kanevski, and W. J. Emery, "Active learning methods for remote sensing image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 7, pp. 2218–2232, 2009.

[43] H. S. Seung, M. Opper, and H. Sompolinsky, "Query by committee," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), p. 287–294, Association for Computing Machinery, 1992.

[44] K. Brinker, "Incorporating diversity in active learning with support vector machines," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 59–66, AAAI Press, 2003.

[45] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," in *International Conference on Learning Representations*, 2018.

[46] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing* (M. Lapata and H. T. Ng, eds.), (Honolulu, Hawaii), pp. 1070–1079, Association for Computational Linguistics, Oct. 2008.

[47] R. Liere and P. Tadepalli, "Active learning with committees for text categorization," in *AAAI*, 1997.

[48] A. McCallum and K. Nigam, "Employing EM and pool-based active learning for text classification," in *ICML*, pp. 350–358, 1998.

[49] S. Tong and D. Koller, "Support vector machine active learning with applications to text classification," in *Journal of Machine Learning Research*, 2002.

[50] L. Huang, "Evaluation of query strategy of active learning for text classification," in *Proceedings of SPIE 12348*, 2022.

[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Association for Computational Linguistics, June 2019.

[52] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "RoBERTa: A robustly optimized

BERT pretraining approach," in *Proceedings of the Association for Computational Linguistics (ACL)*, 2019.

[53] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. A. Smith, "Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping," *arXiv preprint arXiv:2002.06305*, 2020.

[54] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines," in *International Conference on Learning Representations (ICLR)*, 2021.

[55] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[56] OpenAI, "GPT-4 technical report." arXiv preprint arXiv:2303.08774, 2023.

[57] OpenAI, "GPT-4o system card." arXiv preprint arXiv:2410.21276, 2024.

[58] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume* (P. Merlo, J. Tiedemann, and R. Tsarfaty, eds.), (Online), pp. 874–880, Association for Computational Linguistics, Apr. 2021.

[59] T. Goyal, J. J. Li, and G. Durrett, "News summarization and evaluation in the era of GPT-3." arXiv preprint arXiv:2209.12356, 2022.

[60] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, *et al.*, "Evaluating large language models trained on code." arXiv preprint arXiv:2107.03374, 2021.

[61] J. W. Rae, S. Borgeaud, T. Cai, K. Millican, J. Hoffmann, F. Song, J. Aslanides, S. Henderson, R. Ring, S. Young, *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[62] M. A. K. Raiaan, M. S. H. Mukta, K. Fatema, N. M. Fahad, S. Sakib, M. M. J. Mim, J. Ahmad, M. E. Ali, and S. Azam, "A review on large language models: Architectures, applications, taxonomies, open issues and challenges," *IEEE Access*, vol. 12, pp. 26839–26874, 2024.

[63] M. Harman, Y. Jia, and Y. Zhang, "App store mining and analysis: MSR for app stores," in *2012 9th IEEE working conference on mining software repositories (MSR)*, pp. 108–111, IEEE, 2012.

[64] A. E. Hassan, "The road ahead for mining software repositories," in *2008 Frontiers of Software Maintenance*, pp. 48–57, IEEE, 2008.

[65] E. Guzman, M. El-Haliby, and B. Bruegge, "Ensemble methods for app review classification: An approach for software evolution," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 771–776, IEEE, 2015.

[66] F. Rustam, A. Mehmood, M. Ahmad, S. Ullah, D. M. Khan, and G. S. Choi, "Classification of shopify app user reviews using novel multi text features," *IEEE Access*, vol. 8, pp. 30234–30244, 2020.

[67] M. Lu and P. Liang, "Automatic classification of non-functional requirements from augmented app user reviews," in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*, pp. 344–353, 2017.

[68] L. V. G. Carreno and K. Winbladh, "Analysis of user comments: An approach for software requirements evolution," in *2013 35th international conference on software engineering (ICSE)*, pp. 582–591, IEEE, 2013.

[69] N. Chen, J. Lin, S. C. Hoi, X. Xiao, and B. Zhang, "AR-miner: Mining informative reviews for developers from mobile app marketplace," in *Proceedings of the 36th international conference on software engineering*, pp. 767–778, 2014.

[70] M. Viggiato, D. Lin, A. Hindle, and C.-P. Bezemer, "What causes wrong sentiment classifications of game reviews?," *IEEE Transactions on Games*, vol. 14, no. 3, pp. 350–363, 2022.

[71] J. Y. Tan, S. K. A. Chow, and C. W. Tan, "A comparative study of machine learning algorithms for sentiment analysis of game reviews," *Journal of the Institution of Engineers, Malaysia*, vol. 82, no. 3, 2022.

[72] J. Dabrowski, E. Letier, A. Perini, *et al.*, "Analysing app reviews for software engineering: A systematic literature review," *Empirical Software Engineering*, vol. 27, p. 43, 2022.

[73] K. Margatina, T. Schick, N. Aletras, and J. Dwivedi-Yu, "Active learning principles for in-context learning with large language models," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 334–349, 2023.

[74] S. Diao, P. Wang, Y. Lin, R. Pan, X. Liu, and T. Zhang, "Active prompting with chain-of-thought for large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Bangkok, Thailand), pp. 1330–1350, Association for Computational Linguistics, Aug. 2024.

[75] R. Xiao, Y. Dong, J. Zhao, R. Wu, M. Lin, G. Chen, and H. Wang, "FreeAL: Towards human-free active learning in the era of large language models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, (Singapore), pp. 14520–14535, Association for Computational Linguistics, Dec. 2023.

[76] R. Zhang, Y. Li, Y. Ma, M. Zhou, and L. Zou, "LLMaAA: Making large language models as active annotators," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, (Singapore), pp. 13088–13103, Association for Computational Linguistics, Dec. 2023.

[77] H. Rouzegar and M. Makrehchi, "Enhancing text classification through LLM-driven active learning and human annotation," in *Proceedings of the 18th Linguistic Annotation Workshop (LAW-XVIII)*, pp. 98–111, 2024.

[78] SteamCommunity, "Steamworks development: Steam - 2020 year in review," Jan 2021.

[79] Valve Corporation, "Cyberpunk 2077 on steam." `https://store.steampowered.com/app/1091500/`, 2020.

[80] D. Lin, C.-P. Bezemer, and A. E. Hassan, "Studying the urgent updates of popular games on the steam platform," *Empirical Software Engineering*, vol. 22, no. 4, pp. 2095–2126, 2017.

[81] J. Blow, "Game development: Harder than you think," *Queue*, vol. 1, no. 10, pp. 28–37, 2004.

[82] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 8342–8360, Association for Computational Linguistics, July 2020.

[83] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese BERT-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 3982–3992, Association for Computational Linguistics, Nov. 2019.

[84] B. Settles, "Active learning literature survey," Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[85] S. Argamon-Engelson and I. Dagan, "Committee-based sample selection for probabilistic classifiers," *Journal of Artificial Intelligence Research*, vol. 11, pp. 335–360, 1999.

[86] R. Ramadan and B. Hendradjaya, "Development of game testing method for measuring game quality," in *2014 International Conference on Data and Software Engineering (ICODSE)*, pp. 1–6, IEEE, 2014.

[87] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[88] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[89] R. L. Thorndike, "Who belongs in the family?," *Psychometrika*, vol. 18, no. 4, pp. 267–276, 1953.

[90] R. Santos, E. C. Groen, and K. Villela, "An overview of user feedback classification approaches.," in *REFSQ Workshops*, 2019.

[91] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for deep learning in NLP," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[92] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?," in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[93] D. Narayanan, A. Phanishayee, K. Shi, X. Chen, and M. Zaharia, "Memory-efficient pipeline-parallel DNN training," vol. 139, pp. 7937–7947, 18–24 Jul 2021.

Publications

Journal

1. Yejian Zhang and Shingo Takada, "Review Classification Based on Machine Learning: Classifying Game User Reviews," IEEE Access, vol. 11, pp. 142447–142463, 2023.

2. Yejian Zhang and Shingo Takada, "Applying LLMs to Active Learning: Toward Cost-Efficient Cross-Task Text Classification Without Manually Labeled Data," International Journal of Intelligent Systems, 2025.

Conference

1. Yejian Zhang and Shingo Takada, "Lowering the Barrier of Machine Learning: Achieving Zero Manual Labeling in Review Classification Using LLMs," 2025 11th International Conference on Computing and Artificial Intelligence (ICCAI 2025), 2025.