Title	感性情報に基づく機械学習を用いた映画推薦システムのデザイン
Sub Title	Design of movie recommender system with kansei information using machine learning
Author	王, 鳳超(Wang, Fengchao) 小木, 哲朗(Ogi, Tetsurō)
Publisher	慶應義塾大学大学院システムデザイン・マネジメント研究科
Publication year	2020
Jtitle	
JaLC DOI	
Abstract	
Notes	修士学位論文. 2020年度システムエンジニアリング学 第319号
Genre	Thesis or Dissertation
URL	https://koara.lib.keio.ac.jp/xoonips/modules/xoonips/detail.php?koara_id=KO40002001-00002020-0024

修士論文　　2020 年度

感性情報に基づく機械学習を用いた
映画推薦システムのデザイン

王　鳳超

（学籍番号：81933095）

指導教員　教授　小木　哲朗

2021 年 3 月

慶應義塾大学大学院システムデザイン・マネジメント研究科

システムデザイン・マネジメント専攻

Master's Dissertation 2020

Design of Movie Recommender System with Kansei Information Using Machine Learning

Wang Fengchao

（Student ID Number：81933095）

Supervisor　Professor　Ogi Tetsuro

March 2021

Graduate School of System Design and Management,
Keio University
Major in System Design and Management

論　　文　　要　　旨

学籍番号	81933095	氏　名	王　鳳超

論 文 題 目：

感性情報に基づく機械学習を用いた映画推薦システムのデザイン

（内容の要旨）

動画配信サービスの流行に従って、映画を始めとして多くの映像コンテンツをオンデマンドで視聴できる環境が整ってきたが、ユーザの好みは気分によって変化するため、見たいコンテンツを見つけ出すのは容易ではない。感性は人々が映画を選ぶ際に非常に重要な要素である。しかし、現在の推薦システムは、感性による映画の推薦するニーズに満たさない。

本研究では、感性空間上での映画コンテンツの分類を畳み込みニューラルネットワーク（CNN）を用いた機械学習で行い、ユーザの気分に応じた映画コンテンツを感性情報に基づいて検索しユーザに提示する、映画推薦システムについて検討を行った。

訓練済みの機械学習モデルを用い、自動的に映画の感性データを計算することにより、映画感性データベースを構築した。また、ピアソン相関係数で、ユーザの感性要求とデータベース中の映画をマッチングし、近い結果をユーザに提示する。

本研究は Flask を用い、ウェブアプリケーションの開発を行った。結果として、感情による推薦ができるシステムを実現することができた。コンテンツベースフィルタリングを用い、協調フィルタリングのコールドスタート、推薦の不透明、大量のユーザ情報がなければ推薦できないなど問題を避けられた。また、感情による推薦のため、ジャンルなどの映画属性に基づかずに、多様性のある推薦を行うこともできた。

また、検証調査を行い、システムの有効性が検証された。

キーワード（5 語）
機械学習、推薦システム、感性情報、CNN、コンテンツベースフィルタリング

SUMMARY OF MASTER'S DISSERTATION

Student Identification Number	81933095	Name	Wang Fengchao

Title

Design of Movie Recommender System with Kansei Information Using Machine Learning

Abstract

The popularity of video streaming services has created an environment where people can watch movies and many other video contents on demand, but it is still not easy to find the contents they want to watch because users' tastes change according to their emotion. Emotion is a very important factor when people choose films. However, the current recommender systems do not meet the need of recommending movies based on emotion.

In this study, I investigate a movie recommendation system that uses convolutional neural network (CNN) machine learning to classify movie contents in the Kansei space and retrieves movie contents based on Kansei information and presents them to the user according to the user's emotion.

A movie Kansei database is constructed by automatically calculating movie Kansei data using a trained machine learning model. After that, the system matches the user's sensibility request with the movies in the database by the Pearson correlation coefficient, and recommend the close result to users.

In this study, A web application is developed using Flask. As a result, the function of recommending films based on emotions is achieved. By using content-based filtering, the problems of collaborative filtering, such as cold start, lack of transparency, and the need for a large amount of user information, are avoided. In addition, the system can make diverse recommendations, since the recommendation is based on emotions instead of movie attributes such as genre.

The effectiveness of the system was also verified through a validation test.

Key Word(5 words)

deep learning, Convolutional Neural Network (CNN), movie recommendation system, emotional information, Content-based Filtering

目次

図目次

表目次

第 1 章：序論

1.1 研究の背景

映画誕生から百年以上経過した現在、大量の映画コンテンツが存在するため、視聴者はその中から自分が視聴する映画を選択することが容易ではない。特に、動画配信サービスが普及している現在、好きな時間で好きな映像を見られるような環境が整えられつつある一方、自分のみたいコンテンツを見つけることが一層困難になった。

推薦システム（recommender system）は、ユーザにとって有用と思われる対象、情報、または商品などを選び出し、それらをユーザの目的に合わせた形で提示するシステムであるため、情報過多（information overload）の状態から脱する解決策の一つとして普及してきた[14]。

その中、映画の推薦システムは、何を観るか迷っている人にとって、良い解決策となり、多く利用されている。また、推薦システムは映画制作側、動画配信サービス側にとって、観客数、視聴者数を上げることができるメリットもある。そのため、効果的な映画推薦システムの開発は、視聴者、映像提供者の両者にとって価値があると考えられる。

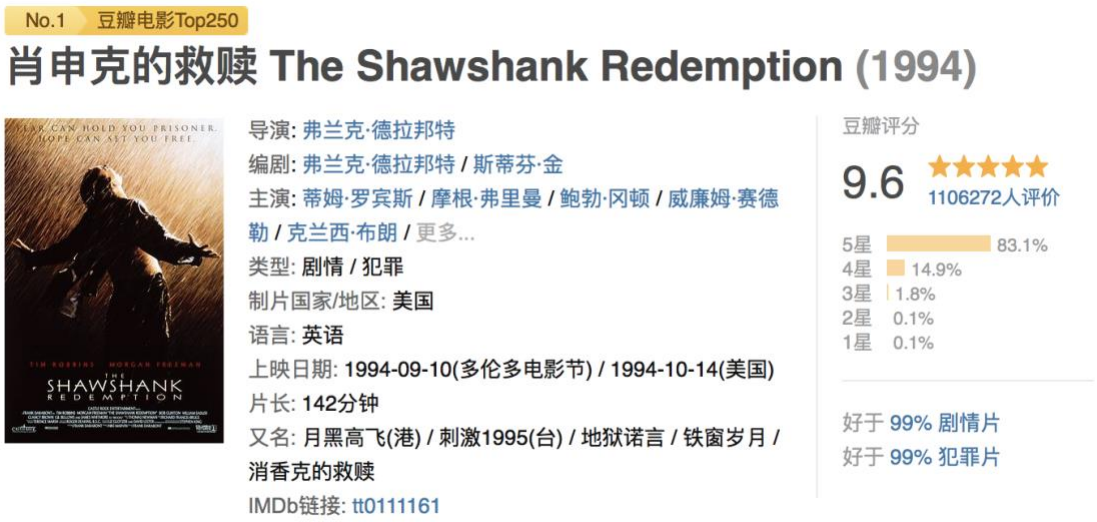


図 1-1 中国の映画推薦システム「豆瓣」における某映画の基本情報

Fig.1-1 Movie Recommender system Dou Ban's web page

しかし、既存の映画推薦システムはユーザにとって充分であると言えるだろうか。現在利用されている代表的な映画推薦システムには、アメリカの IMDb、中国の豆瓣（図 1−1）、そして動画配信サービス Netflix[2]などがある。

典型的な映画推薦システムで提供される情報は、映画のタイトル、ポスター、監督、キャスト、ジャンル、そして、５つ星のランク（映画の総合的な良さ）などの基本的な情報の部分と、星のランク、文章で構成されているユーザレビューの部分で構成される（図 1−2）。

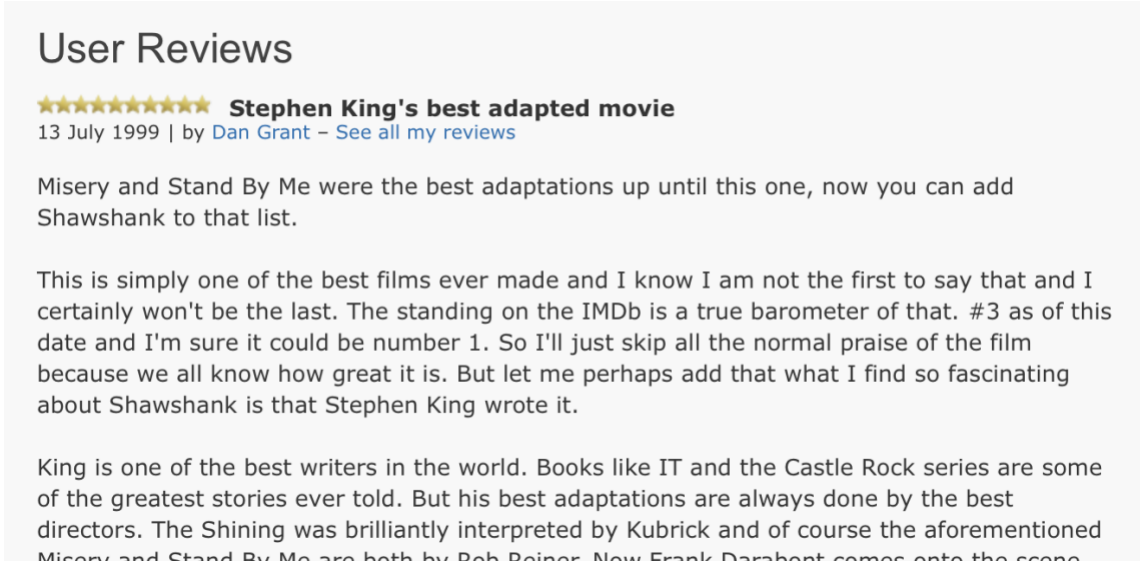


図 1-2 IMDB ユーザレビュー

Fig.1-2 A user’s review on IMDB

また、レコメンデーションの方法としては一般に、利用者の行動履歴に基づいて推薦を行う協調フィルタリングと、コンテンツの属性に基づいて推薦を行うコンテンツベースフィルタリングの 2 つの方法が、広く用いられている。

しかしながら、ユーザの好みは気分によって変化するため、その時に観たいコンテンツを見つけ出すのは容易ではない[3]。

特に多くの映画推薦システムで使用されている協調フィルタリングの方法は、多くの利用者からの閲覧履歴に基づいた推薦を行うため、人気がないために閲覧履歴のない映画、または人気があっても閲覧履歴のない新規映画が推薦できないという問題点がある。また、利用者の状況や気持ちが変わっても推薦する映画は変化しない。多くの人が映画を選ぶ際に重視している要素に応じて推薦するニーズを満たすことができない。

1.2 映画推薦システムの現状と問題点

天気、気分、目的などの「状況」により、人々の感性・気持ちが変化している。映画を選択する際、その感性・気持ちの変化により、映画に対する好みも変化している。そのため、見たい映画が異なる。

例えば、仕事が終わって、ストレスが溜まっているとき、楽しくて、軽い映画を見たい傾向がある。一人きりで、静かにしたい場合、ちょっと悲しい映画を観たくなることもある。

映画は人々の感性を簡単に喚起するメディアであり、映画を選ぶ際に、感性情報は無視してはいけない要素の一つだと考えられる[6]。

しかし、現在の推薦システムの映画基本情報の部分（図 1−1）ではこのニーズを満たせない。図 1-2 のように、ユーザレビューの部分は映画について様々なスタンスからの評価があるので、感性情報を提供する可能性があり、このニーズを満たすかもしれない。だが、ユーザレビューは観点がバラバラで主観的な意見である。そのため、ユーザレビューの信頼性や可用性に問題があると考えられる。さらに、ユーザレビューは文章の形のため、欲しい情報を抽出しにくく、情報を利用しにくいというデメリットもあると思われる。

動画配信サービスでのレコメンデーション機能はどうだろう（図 1-3）。現在、動画配信サービスの流行に従って、映画や動画推薦システムも多く使われている。例えば、DVD レンタルサービスから動画配信の事業を展開してきた Netflix をはじめ、Amazon Prime Video、Hulu など。

既存の映画推薦はまず、場合、時期による気持ちが変化するにもかかわらず、推薦してくれる映画は変わらない。つまり、このニーズを満たせない。

また、現在の映画推薦システムはほぼ協調フィルタリングを使用している。そのため、協調フィルタリングの欠点も問題点となっている。

図 1-3 Amazon Prime Video のレコメンデーション

Fig.1-3 Amazon Prime Video's recommendation

その一つは、コールドスタート問題（Cold Start Problem）である。協調フィルタリングを利用する際に、大量のユーザ情報、閲覧履歴が必要である。

そのため、まず、新規映画は、ユーザの鑑賞履歴がないので、推薦することが難しい問題点がある。また、人気の少ない映画は、ユーザの鑑賞履歴や評価が少ないため、評価することも難しい。さらに、新規ユーザは閲覧履歴がないので、推薦することも困難である。

また、現在の映画推薦システムは、同じカテゴリーの映画を推薦する傾向が高くて、ユーザの観る映画の種類が限定的になる可能性がある問題点もある。

1.3 研究の目的

本研究ではこのような問題に対し、利用者の気分や感性を取り入れた映画コンテンツの推薦システムを実現することを目的とする。

そのため、感性空間上での映画コンテンツの分類を畳み込みニューラルネットワーク(CNN)を用いた機械学習で行い、ユーザの気分に応じた映画コンテンツを感性情報に基づいて検索しユーザに提示する方法について検討を行った。

すなわち、ユーザの気分による好みの変化を踏まえたレコメンド機能を実現する。またここで用いる推薦方法は、コンテンツベースフィルタリングになるため、他者の人気に関係なく個人の好みに応じたコンテンツを推薦できる点が特徴である。

1.4 既存の推薦方法

現在、多くの推薦システムは映画推薦に限らず、オンライン販売、音楽サービス、画像検索など様々な領域で活躍している。

例えば、Amazon.com はユーザの購買履歴に基づき、おすすめ商品を提示する、Reddit はニュースなどを推薦するなど。

多く使われている推薦方法は、協調フィルタリングとコンテンツベースフィルタリングなどがある。以下で説明する。

1.4.1 コンテンツベースフィルタリング

コンテンツベースフィルタリング（内容ベースフィルタリング、Content-based Filtering)は事前にアイテムを属性に基づいて、グルーピングをして、ユーザが検索や購入したアイテムと似たアイテムを特徴が似ている順に表示し、推薦を行う方法である。

コンテンツベースフィルタリングの特徴はアイテムを根拠として推薦する。この実現のためには、アイテムの属性を分析し、構造化したデータでデータベースを作る。ユーザの履歴や要求により、データベース中の似たようなアイテムを推薦する。

コンテンツベースフィルタリングの原理により、ユーザの独立性（他のユーザに影響されないこと）があり、推薦の透明性（transparency、ユーザに推薦する根拠をはっきり説明できること）、新しいアイテムや人気のないアイテムもすぐ推薦できるなどの長所がある。

しかしながら、構造化できない複雑な属性を持つアイテムに対しては、データを構造化しにくいため、コンテンツベースフィルタリングで推薦することが

困難である。またユーザ履歴を参照し、推薦するシステムは、履歴と同じカテゴリのアイテムだけを推薦し、ユーザの潜在嗜好を発見できない（over-specialization）などの短所がある。

1.4.2 協調フィルタリング

協調フィルタリング（Collaborative Filtering）の特徴は、未知のアイテムの有用度推定において、他ユーザの意見（既知アイテムに対する有用度評価）を利用する点である[1]。つまり、多くのユーザの情報を使用履歴から蓄積し、あるユーザに類似した他のユーザの嗜好・情報を用いて、自動的に推論を行う方法である。

例えば、Amazon.com の「おすすめの商品」は協調フィルタリングを用いている。同じ商品を買ったユーザ（類似ユーザ）の購買歴から、自動的にこのユーザが買う可能性の高い商品を推薦する機能である（図 1-4）。

図 1-4 Amazon.com のおすすめの商品機能

Fig.1-4 Amazon.com's recommendation products

協調フィルタリングの原理により、以下の長所がある。推薦するアイテムの情報や分析が要らないため、推薦対象が複雑でも推薦できる。アイテムの属性情報を根拠にしていないので、意外性のある結果を推薦することもある。

しかし、協調フィルタリングを有効にするためには、大量のユーザ情報が必要である。そのため、コールドスタート（cold-start）という課題は指摘されている。

コールドスタート（cold-start）とは、大量のユーザ行動履歴・情報に依頼しているため、新規アイテムや、人気のないアイテムにユーザの行動履歴がないまたは少ないため、推薦しにくい問題である。そして、新規ユーザまたは利用する頻度の少ないユーザに対し、推薦することが困難である。

また、システムの操作履歴（例えばブラウザの閲覧履歴）などを利用した暗黙的なものであるため、推薦の透明性はよくないなどの短所がある。

1.5 推薦システムにおける挑戦と推薦方法の比較

（１）複雑な属性を持つアイテムの推薦

アイテムの属性は構造化（structured）データにできるものと構造化できない（unstructured）ものに分類することができる。構造化属性は明確であり、数値化しやすい特徴がある。非構造化属性は意味が曖昧で、数値化しにくく、直接に使うことが困難である特徴がある。

人間を例にすれば、構造化データにできる属性は名前、年齢、身長などは構造化データであり、簡単且つ正確に抽出できる情報である。その一方、人間の嗜好、価値観などは非構造化属性である。

映画や音楽などのメディアは構造化属性がある一方、非構造化属性も多くある。例えば、映画は、タイトル、監督、年代などの構造化属性を持ちつつ、ストーリー、展開のテンポ、雰囲気などの非構造化属性も多く持っている。

コンテンツベースフィルタリングは、アイテムの属性を根拠として推薦を行うため、非構造化属性を持つアイテムが推薦しにくい短所がある。Amazon プライムビデオなどは、簡単な属性（ジャンル、年代、国）で、コンテンツベースフィルタリングで映像を推薦しているが、

協調フィルタリングは、大量のユーザ活動履歴だけに基づく推薦を行うので、アイテムの属性情報を分析する必要がなく、これが現在の映画推薦システムがほぼ協調フィルタリングを使用している要因である。

（２）コールドスタート問題と少数者推薦

協調フィルタリングに代表される推薦アルゴリズムは、評価データの蓄積が乏しい新規ユーザや新規アイテムの推薦が難しいとされるコールドスタートという問題を本質的に抱えている[15]。

しかし、Web サービスにおけるユーザやアイテムの流動性がますます高まるにつれて、その問題に対処するという現場でのニーズは非常に大きい[26]。

新規映画の増える速度が遅くない。統計局より、2019 年日本における映画館で公開された映画は合計 1278 本であった。世界中で、ものすごい速さで新規映画が増えていることが推測できる。そのため、コールドスタート問題の解決は映画推薦システムにとって非常に重要だと言える。

コンテンツベースフィルタリングは、アイテムに基づき推薦するので、コールドスタートの問題を避けることができる。

また、コンテンツベースフィルタリングは、人気のないアイテムも推薦でき、しかも、ユーザの独立性があり、他のユーザに影響を受けない。そのため、嗜好の少数者を対象として、推薦する場合は協調フィルタリングより良い効果をもらえると考えられる。

（３）新規性（novelty）多様性（diversity）を踏まえる推薦

推薦システムに対して、推薦の新規性・多様性が重視すべきである。

個性を重視する推薦が進みすぎた結果、フィルターバブル（filter bubble）により多様性が失われるという潜在的な問題がますます注目を集めている[15][28][27]。一方、ユーザの視点から見ると、新規性や多様性は満足度の直接な理由として、望ましいものである。消費者行動についての研究では「新規性、

意外性、変化、複雑性」（novelty, unexpectedness, change and complexity）から満足感を得るという結論もある[30]。

また、新規性、多様性のある推薦はユーザの体験を豊かにし、視野を広げることもできる[15]。

映画推薦システムの場合、多様性を重視する映画推薦システムにおいて、ユーザに多様性のある映画を推薦することによって、ユーザの視野を広げ、映画鑑賞能力を向上させることも期待できる。ユーザの映画に対する興味を保ちつつ、また推薦システムを利用し、映画を見る良い循環になれると思う。

従来のコンテンツベースフィルタリングを用いた映画推薦システムは、映画の抽出しやすい属性を使い、映画の分類を行う。例えば、映画のジャンル、国別、公開された年代など抽出・利用しやすい属性を使い、映画を分類する。その上、ユーザの閲覧履歴から、同じ属性の映画コンテンツを推薦する。したがって、ユーザに推薦する結果が徐々に収束し、同じ種類のコンテンツを推薦する傾向があり、ユーザの見る映画の種類が単一化になるリスクが高いと指摘されている。

その一方、協調フィルタリングはコンテンツのジャンル・作者・国などを根拠とせずに、推薦を行うため、コンテンツベースフィルタリングより、新規性・多様性のあるアイテムを推薦することができる。

1.6 関連研究

（１）映画の感性分類についての関連研究

映像は簡単に人の感情を喚起させる特徴があるメディアである。映像メディアによる感情についての研究も多くされている。

Philippot（1993）は、フランス人を対象として、12 種類の映像を見せる実験を実施した。映像により生起する 6 つの感情 : 幸福（happiness）、怒り（anger）、嫌悪（disgust）、恐怖（fear）、中立（neutral）、悲しみ（sadness）を明らかにした[8]。

また、Gross と Levenson（1995)は Philippot（1993）の研究を基に、再生時間、内容の明瞭性と複数の感性を起こさない離散性の三つの基準を設定した上で、実験を行った。実験結果を分析し、楽しさ（amusement)、怒り（anger)、満足（contentment)、嫌悪（disgust)、恐怖（fear)、中立（neutral）、悲しみ（sadness）、驚き（surprise）の 8 種類の感情が生起するという Philippot（1993）の研究と類似した結果を明らかにした[10]。

Zhao 等の研究では芸術写真の、芸術に基づく感情特徴（Principles-of-art-based emotion features ,PAEF）を抽出し、機械学習を用い、芸術画像を Amusement, Awe, Contentment, Excitement, Anger, Disgust, Fear, sadness の８感情による分類している[5]。

（２）メディアの定量的評価に関する研究

映像、または音楽などの定量的評価方法を探る研究もある。映像の定量化評価ができると、映画をコンテンツベースフィルタリングでより正確に推薦することが可能である。

従来の映像・音楽の定量的評価を行う手法として、SD 法などはよく使われている：

高橋靖（2000）は映画内容を定量的に評価・解釈するために、映画における物語の意味的単位であるシーンを複雑化･解決化の尺度で評価するセマンティックスコア法を提案した[11]。

池添剛ら（2001）は SD 法並びに因子分析により感性空間を用い、感性語による音楽検索システムをデザインした。映画の定量的評価の標準を設定する際に、本研究の参考となると考える[12]。

コンテンツベースフィルタリングで映画を推薦する困難さは、どうすれば映画の属性を抽出し、ユーザと繋げることができるかという点だと考えられる。そのため、映画の定量的評価の研究は本研究の参考になると思われる。

（３）従来の感性工学研究

従来の感性工学の研究は主に図 1-5 に示している手法を使い、感性工学の研究を行っている。

具体的には、まず SD 法を使い、研究対象の感性空間を測る。そして、研究対象の物理属性をできるだけ抽出する。最後に、測った感性と物理属性の間の関係性を因子分析などの統計的手法で見つける[8]。

図 1-5 従来の感性工学研究手法

Fig.1-5 Conventional Kansei Engineering Research Methods

しかしながら、SD 法を用いた従来の感性工学は直接に映画の感情分類に利用することが困難である。原因は、映画は物理属性が多くて、複雑である。例えば、映画の画面（色彩、構図など）、物語（展開、テンポなど）などの属性は複雑であり、データ化しにくい属性が多い。その上、感性と物理属性の関係性を見つけることも困難である。

1.7 本研究の新規性

（１）感性による映画推薦ニーズに満たす

映画は人の感性を簡単に喚起できるメティアとして、人々は映画を選ぶ際に、場合、一緒に見る人、時期などの要素による気分や、感性の変化による映画の好みの変化が重要だと考えられる。つまり、ユーザの好みは変わらない物ではない。しかし、従来の映画推薦システムは、ユーザの好みを変わらないものとして捉え、映画を選ぶ際、重要な要素である、感性を反映していないという課題がある。

本研究は、まず、映画に対し感性による分類を行うことが新しいところである。また、感性による映画推薦の提案・システムもユーザのニーズを満たすことができる。

（２）非構造化属性によるコンテンツベースフィルタリングの問題解決

現在、多く使われている映画推薦システムはほぼ協調フィルタリングを利用している。その理由は、コンテンツベースフィルタリングは非構造化属性を持つ複雑なアイテムを推薦することが困難であるという点である。映画推薦システムでは、映画に関しても、ユーザ（人間）に関しても、数値化しにくい属性が多くあり、また、人間と映画の繋がりを見つけることも困難である。

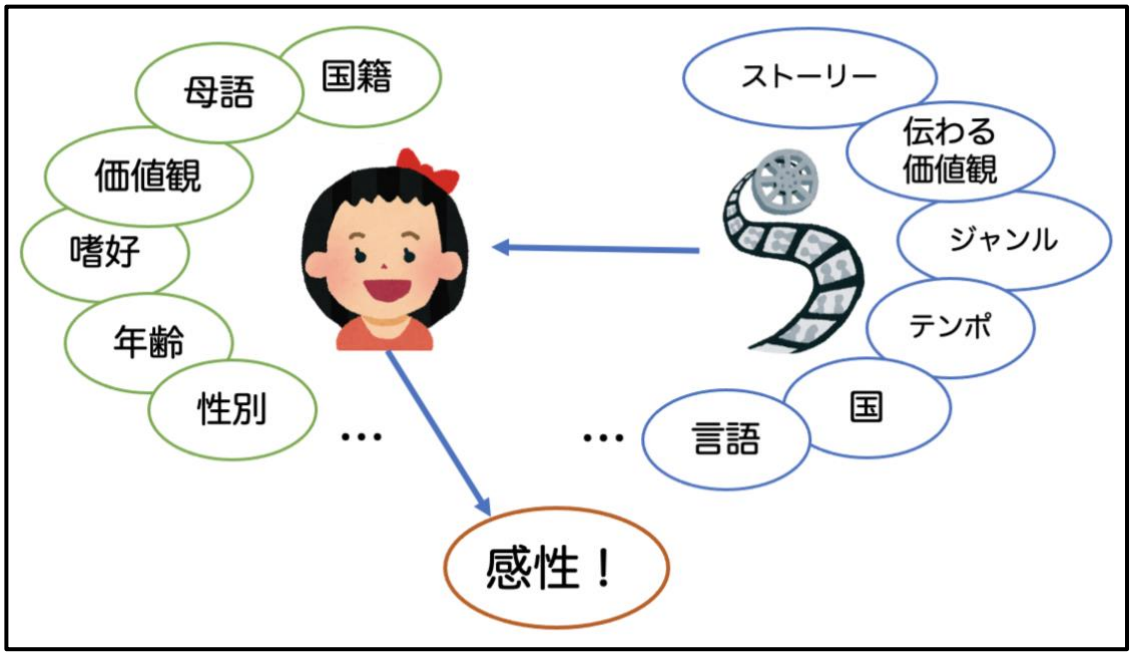


図 1-6 感性で人と映画をつなげる

Fig.1-6 Connect people with movies by emotion

本研究の特徴は、これまでのコンテンツベースフィルタリングでは映画などの複雑な非構造化属性を持つアイテムを推薦するのが困難であった課題に対する解決策だと考える。

ここで用いるコンテンツベースフィルタリングでは、映画や人間の独自の属性を分析するのではなく、映画が視聴者生起させる感情を直接的に数値化し分類を行う。感情は映画から生起させられるので、自然にユーザと映画を繋げ、推薦することもできる。しかも、音楽や映画などのメディアでは、同じ作品によって生起される感情は違う人でも共通性があり[12][31]、ユーザの個性を踏まえなくても、正確に推薦することが可能である

（3）コンテンツベースフィルタリングを用いた映画推薦

本研究は、映画を感性によって分類を行うことで、コンテンツベースフィルタリングを利用し、推薦システムを構築した。この方法は、協調フィルタリングのコールドスタート問題を解決することができ、またユーザへの依存性を減らすことができる。

（4）映画推薦の多様性を保つ

一般に、推薦システムの普及及び推薦の正確さが徐々に上がるに従って、人々の接触する情報の範囲が狭くなっていく。

しかし、推薦システムの推薦方法を調整することにより、ユーザの接触する情報の範囲は変わっていくと考えられる。そのため、本研究は、映画を推薦すると共に、ユーザが幅広い映画を見る可能性を向上させることの実現も目指す。

第 2 章：感性情報に基づく機械学習を用いた

映画推薦システムの提案

2.1 概要

本章では、感性情報に基づく映画推薦システムの提案内容とシステムの全体設計について説明する。

2.2 提案内容とシステム全体設計

（1）目標の確定

本研究は映画を選ぶ際に、感情という嗜好が影響する重要な要素に着目し、既存の映画推薦システムが満たせない感性情報による映画推薦システムの実現を目的としている。

そのため、本システムの開発においては、まず、感性情報に基づく映画の推薦という機能を実現しなければならない。

また、現在の映画推薦システムが利用している協調フィルタリングによるコールドスタート、推薦の不透明さ、大量のユーザ行動履歴がなければ推薦できないなどの問題を解決することを目指す。

さらに、ユーザの映画鑑賞の視野を広げるために、多様性のある推薦をすることを目指す。

（2）システム設計の方向性

以上の目的を実現するには、まず、協調フィルタリングでなく、コンテンツベースフィルタリングを使用する。コンテンツベースフィルタリングを使用することにより、協調フィルタリングのコールドによるスタート、推薦の不透明さ、利用者への依存などの問題を解決することができる。

コンテンツベースフィルタリングでの映画推薦の困難さは、非構造化属性を多く持つ映画の属性分析と、映画と同様に複雑なユーザ（人間）との繋がりを見つけることである。

そのため、推薦する根拠を、映画が視聴者に生起させる感情に決定した。直接に映画と人間の独自の属性を分析し、マッチングするのではなく、映画が視聴者に生起させる感情を数値化し、映画の分類を行う。

感情は映画から生起されるので、自然にユーザ（人間）と映画を繋げ、推薦することもできる。また、音楽や映画などのメディアは同じ作品によって生起される感情は違う人でも共通性があり[12][31]、ユーザの個性を踏まえなくても、ある程度正確に推薦することが可能である。

感性による映画の分類は、機械学習で行う。その理由は、新規映画も推薦できるようにするため、新規映画を自動的に分類することが必要である。また、映画の各物理属性（色彩、テンポなど）は感情喚起に関係があるが、その関係が複雑であり、計算しにくい。映画の画像に対し、生起される感情による分類を行う機械学習モデルを訓練して保存すれば、新規映像を自動的に分類することが可能である。

映画を分類するためには、画面、音楽、セリフ、ストーリー、テンポなど映画の各構成要素が存在するが、この中で最大限に映画の特徴を表すのは映画の

画面である。また、画像の特徴を学習し、分類を行う機械学習技術は発展してきており、CNN（Convolutional neural network、畳み込みニューラルネットワーク）というディープニューラルネットワークなどの精度の高い学習モデルが作られている。

（３）全体設計

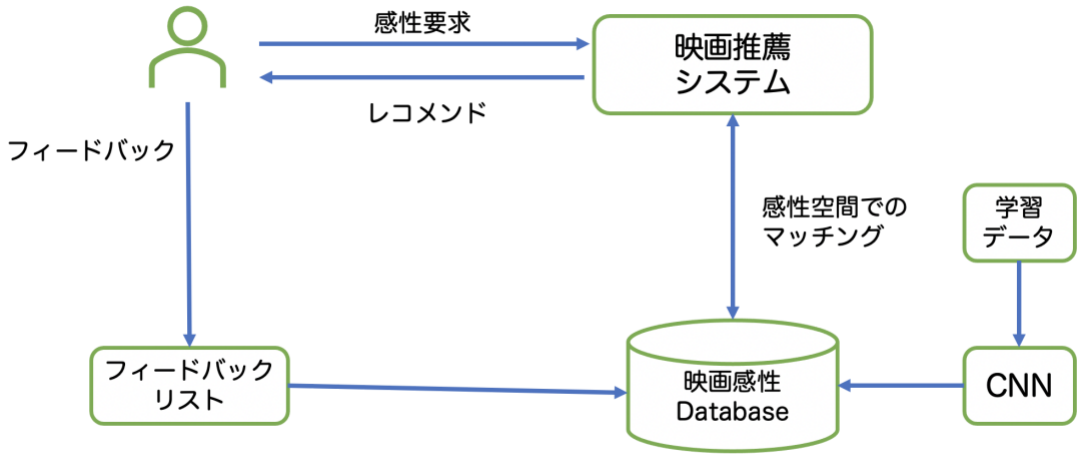


図 2-1 システム設計図

Fig.2-1 system design diagram

上述の方向性を踏まえて、システムの全体設計について説明する。

本研究は、機械学習による感性分類に基づき、ユーザの気分や感性を踏まえた新しい映画推薦システムの構築を行う。図 2-1 は、システムの全体構造を示している。

ユーザが映画推薦システムに感性要求を渡し、映画推薦システムがユーザの感性情報に基づく推薦結果を返す。

この推薦結果は、映画感性データベースから検索し、ユーザの感性に近いものを抽出する。

映画感性データベースの構築は、訓練済みの機械学習モデルを用い、自動的に多数の映画の感性情報を予測する。

具体的には、映画から抽出した画像に感性ラベルをつけて、学習データを作る。CNN を用い、この学習データで訓練を行う。このように、学習モデルを作ることができる。

また、推薦の調整や学習の調整をするために、フィードバック機能も加える必要がある。

感性に基づく映画推薦システムは、次のサブシステムから構成される。

□　映画の感性データベース

感性情報に基づく映画推薦システムでは、まず映画のコンテンツ映像から自動的に画像をスクリーンショットとして抽出し、多量の画像を収集する。サーバ上ではこれらのデータに対して、訓練済みの機械学習モデルで各映画の感性を判断し、記録する。

□　感性に基づく映画を推薦する機能

本システムでは、ユーザが映画を検索する際には、映画によって生起される具体的な感情およびその感情に関わるシチュエーションなどをキーワードとして提示し、ユーザはそのときの自分の状況にあったキーワードを選択し、値を入力させることで、ユーザの感性を抽出する。

ユーザの感性情報を収集し、データベースからユーザに応じた映画を検索し提示することで推薦を行う。この際、ユーザのフィードバックを収集する機能を想定しておくことで、運用しながら機械学習の精度を上げることが期待できる。

第 3 章：機械学習の方法

3.1 概要

本章では、映画を感性によって分類するために、構築した機械学習モデルについて説明する。

使用した機械学習プログラムは、畳み込みニューラルネットワーク（CNN)を用いた。機械学習のオープンソースソフトウエアライブラリ TensorFlow、およびニューラルネットワーク用ライブラリ Keras を用いて実装を行った。Keras では、シーケンシャルモデルを使用して画像データセットに対する各処理および学習を実行する。すなわち、CNN モデルの各レイヤーを単純に積み重ねていくことで、画像の入力から、畳み込み演算、活性化関数、プーリング、分類の各処理が連続的に行われる。

3.2 使用したツールの紹介

（１）TensorFlow と Keras について

TensorFlow（テンサーフロー）は、Google が開発しオープンソースの、機械学習に用いるためのソフトウェアライブラリである。

Keras（ケラス）は、Python で開発されたオーペンソースニューラルネットワークライブラリである。TensorFlow、Microsoft Cognitive Toolkit、Theano などの上で、動作することができる。Keras はライブラリ TensorFlow のインタフェースとして機能している。

Keras には、レイヤー、目的、活性化関数、オプティマイザーなど、一般的に使用されるニューラルネットワークビルディングブロックの多数の実装と、ディープニューラルネットワークコードの記述に必要なコーディングを簡素化するための画像およびテキストデータの操作を容易にする多数のツールが含まれている。

標準のニューラルネットワークに加えて、Keras は畳み込みニューラルネットワークとリカレントニューラルネットワークをサポートしています。ドロップアウト、バッチ正規化、プーリングなどの他の一般的なユーティリティレイヤーをサポートする。

（２）畳み込みニューラルネットワーク（CNN）について

畳み込みニューラルネットワーク（Convolutional neural network、たたみこみニューラルネットワーク、略称 : CNN）は、フィードフォワードニューラルネットワーク（Feedforward neural network、順伝播型ニューラルネットワーク）の一種である。画像や動画認識に広く使われている機械学習モデルである。

フィードフォワードニューラルネットワークは、最初に考案された最も単純なタイプの人工ニューラルネットワークである。 このネットワークでは、情報は、入力ノードから隠れノード（もしあれば）を経て、出力ノードに向かうという一方向にのみ移動する。ネットワークにはサイクルやループはない（図 3-1）。

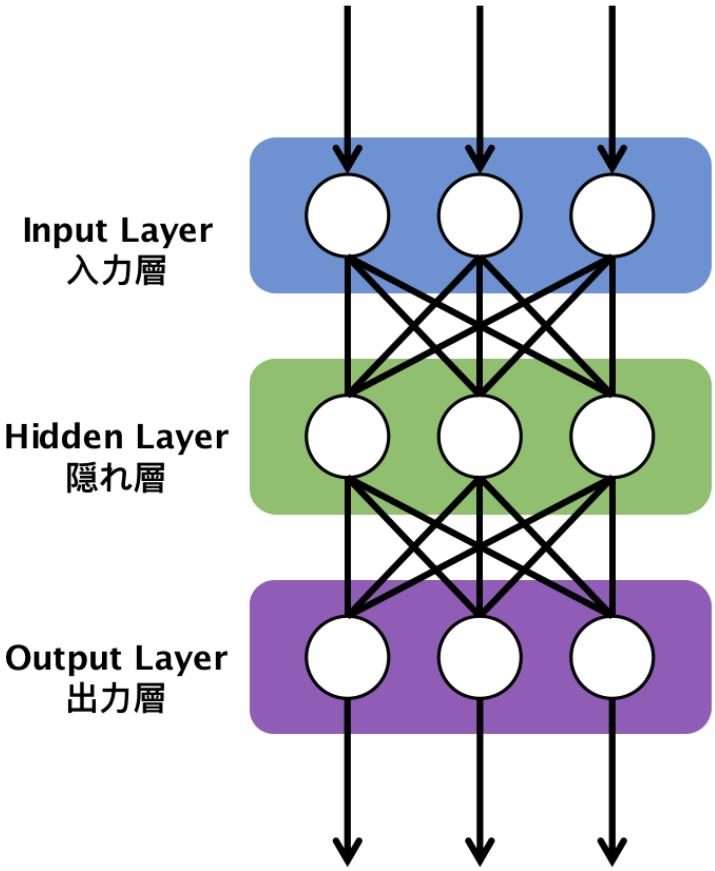


図 3-1 フィードフォワードニューラルネットワーク構造のイメージ

Fig.3-1 A feedforward neural network architecture image

畳み込みニューラルネットワークは、入力層（input layer）、隠れ層（hidden layer）、および出力層（output layer）で構成されている。

3.3 使用した機械学習モデル

（１）畳み込み層

CNN の畳み込み層は、画像の特徴を抽出する役割を果たす。畳み込み層の計算では、入力画像のデータサイズに対し、各畳み込み層後のデータサイズ heightout, widthout は以下の式で表される。ここで、heightin, widthin は入力データサイズ、heightkernel, widthkernel はカーネルデサイズ、padding、stride はそれぞれパディング、ストライドの値である。

$$height_{out} = (height_{in} - height_{kernel} + 2 \times padding) \div stride + 1$$

$$width_{out} = (width_{in} - width_{kernel} + 2 \times padding) \div stride + 1$$

ここでは、第 1 層はカーネルサイズを 3x3、フィルタ数を 16、パディングを 0、ストライドを 1、第 2 層はカーネルサイズを 3x3、フィルタ数を 32、パティングを 0、ストライドを 1、第 3 層から第 5 層まではカーネルサイズを 3x3、フィルタ数を 64、パディングを 0、ストライドを 1 としたモデルを用いて計算を行った。

活性化関数としては ReLU を使用した。判定プロセスは、入力値が 0 未満の場合、出力値はすべて 0 であり、0 より大きい場合、出力値は入力値と等しくなる。これにより、勾配の消失による情報損失を回避し、モデルの学習効率を向上させることができる。

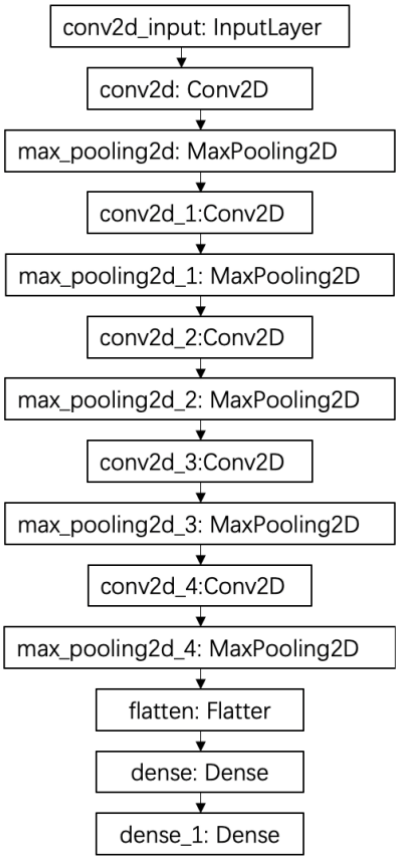


図 3-2 ネットワーク図

Fig.3-2 Network diagram

（2）プーリング層

プーリング層では最大プーリングを使用し、2x2 のブロックに分割し、最大値を選択することで画像の特徴を維持しつつデータの縮小を行った。

図 3-2 に、本研究で使用した CNN のニューラルネットワークモデルを示す。図 3-2 に示す通り、使用したモデルでは、畳み込み、プーリング層が 5 層、全結合層が 2 層、合計 7 層のネットワークを構築した。

3.4 映画を感性による分類の機械学習実験

本研究では、上述の CNN のニューラルネットワークモデルを使用し、映画コンテンツの分類に関する機械学習を実施した。映画の学習用データとしては、予告編映像を対象として使用した。予告編映像は映画の宣伝として使用されるため、映画の物語や雰囲気を効率的に表現した映像素材で構成されていると考えられる。本研究では、視聴者の種々の感性に対応する 100 本の映画コンテンツを選択し、それぞれの予告編映像からスクリーンショットで得られた静止画像を機械学習の画像データセットとして使用した。

映像から受ける感性に関しては、Philippot（1993）は、映像により生起する６感情：幸福（happiness）、怒り（anger）、嫌悪（disgust）、恐怖（fear）、中立（neutral）、悲しみ（sadness）を明らかにした[8]。また、Gross と Levenson（1995)は Philippot（1993）の研究を基に、楽しさ（amusement)、怒り（anger)、満足（contentment)、嫌悪（disgust)、恐怖（fear)、中立（neutral）、悲しみ（sadness）、驚き（surprise）の 8 種類の感情が生起することを明らかにした[10]。Zhao 等の研究では芸術写真の、芸術に基づく感情特徴（Principles-of-art-based emotion features ,PAEF）を抽出し、機械学習を用い、芸術画像を Amusement, Awe, Contentment, Excitement, Anger, Disgust, Fear, sadness の８感情による分類している[4-5]。本研究では、上述の感情分類を参考にし、映画により多く生起する感情として、楽しさ（amusement）、安心(contentment)、嫌悪(disgust)、恐怖(fear)、怒り(anger)、悲しみ(sadness)、興奮(excitement)の７クラスを取り上げた。実験用に選択した 100 本の映画コンテンツに対し、実験者の主観によりそれぞれを７つの感性カテゴリに分類し、実験に使用した。

```
Epoch 500/500
5/5 [==============================] - 14s 3s/step -
loss: 0.2187 - accuracy: 0.9299 - val_loss: 0.0645 -
val_accuracy: 0.9788
```

図 3-3 機械学習の精度

Fig.3-3 Accuracy of the deep learning model

機械学習による分類方法としては、まず 100 本の映画の予告編映像からそれぞれ 10 枚以上の画像をランダムに抽出し、300x300 のデータサイズにした画像を使い、機械学習を実行した。機械学習においては、用意した画像のデータセット 4106 枚のうち、学習に 2874（70%）枚、テストに 1232（30%）枚を使用した。当初、学習収束後の CNN モデルに対する、テストデータの分類結果

の判定精度は非常に低かった。そのため、精度を向上させるために、画像データセットの整理と関数を直す二つの方向で機械学習精度のマネジメントを行った。結果として、精度を 90%以上に向上させた。

Keras の Model. save ( ) を使い、訓練済みの学習モデルを保存した。

3.5 学習精度のマネジメント

映画予告編から抽出した画像を機械学習で感性による分類を行う実験の学習精度は最初 14%で、ランダムに判断するほど、学習精度としては非常に低かった。機械学習の精度を向上させるために、画像データセットの整理と関数を直す二つの方向で機械学習精度のマネジメントを行った。

まず、画像データセットについて、三つの原因が推測される。第一は、映画の予告編映像からランダムに得た画像には、リリース日、登場人物の紹介などの画像も含まれており、これらは機械学習にとっては、ノイズ情報である。第二は、学習に使用した画像データの数が低いことが考えられる。また第三は、使用した画像データは予告編映像からランダムに抽出されたため、必ずしも感性情報を表していないシーンが含まれていたり、反対に分類に必要なシーンが含まれていなかったことが考えられる。

そのため、学習精度を向上させるために、以下の３通りの方法を用いて、使用する画像データセットの修正を行った。

（１）ノイズ画像の除去

映画予告編映像からランダムに切り出して得た画像データセットから、人工的に、リリース日、登場人物の紹介、真っ黒な画像など、ノイズデータと考えられる画像を外して、同様の機械学習を実行した。

その結果、学習収束後のテストデータに対する判定精度は 21%となった。この結果は、ランダムに抽出した画像をそのまま使用した場合に比べては向上したが、判定精度としてはまだ低いことが分かる。

（２）学習データの追加

Jana Machajdik 等は、芸術写真を用いたデータセットに対して感情分類の機械学習を行い、60%以上の判定精度を示している[6]。ここでは、学習データの画像の数を増やすため、Machajdik が使用した芸術写真のデータセット（合計 807 枚)を映画予告編から得た画像データに加えて、機械学習を実施した。

この結果、学習収束後の判定精度は 16%であった。ここでは、学習データの数を増やすことにより判定精度が高くなることが期待されたが、結果は反対に学習精度が低下した。その理由としては、芸術写真はアーティストが意図的に表現とデザインテクニックを使い、適切な雰囲気を作った画像であり、映画画面の表現とは異なっていることが推測されている。また、画像の中で人間が含まれる割合が映画と比べると少なく、両者の画像の特徴が異なっていたことが原因と考えられる。

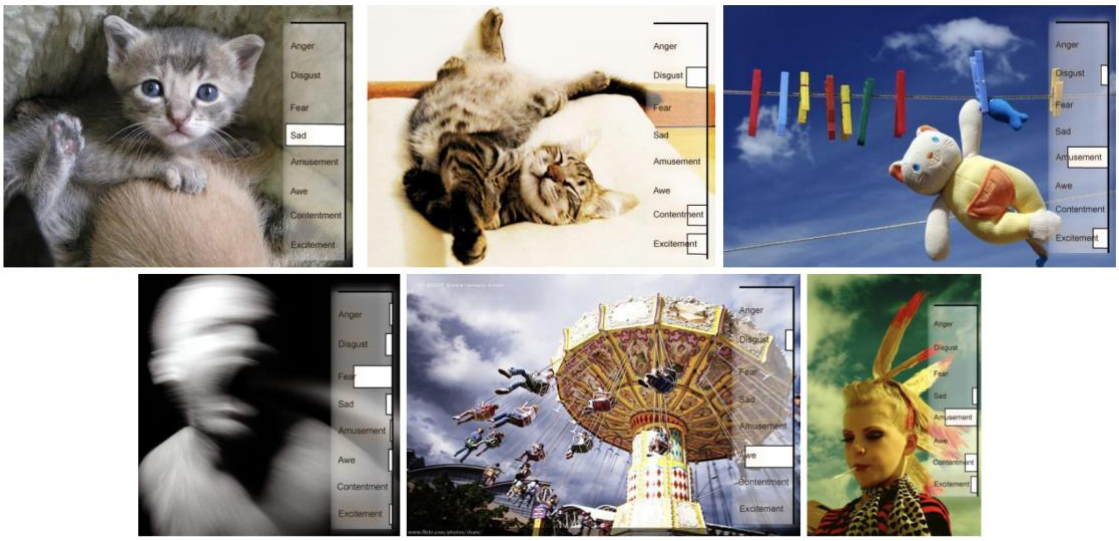


図 3-4Machajdik の芸術写真を用いた感情分類データセット

Fig.3-4 Affective image classification datasets using artistic photography by Machajdik

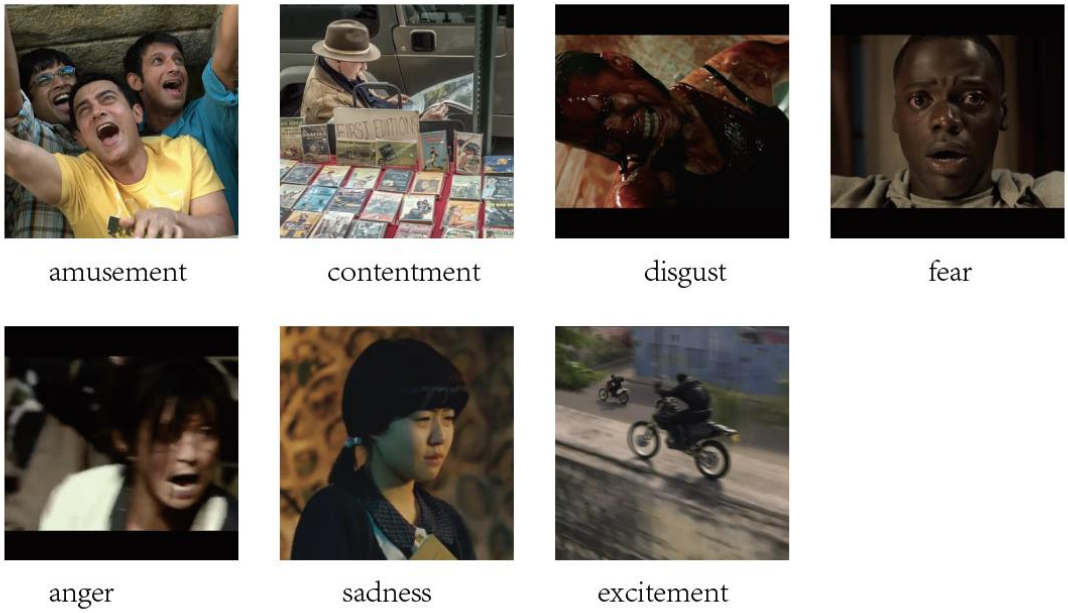


図 3-5 本研究で構築された映画画面を用いた感情分類データセット

Fig.3-5 Emotion classification dataset using movie screens built in this research

（３）典型的な感性シーンの抽出

次に、機械学習における感性分類の精度を向上させるため、典型的な感性シーンの画像を人為的に選択する方法を用いた。具体的には、映画予告編映像からランダムに抽出された画像データセットから、曖昧な画像を除去し、人間の目で確認しながらより明確に感情を判断できる画像を抽出し、学習データとして使用した（合計 1277 枚）。

この結果、学習収束後の判定精度は 32%であり、ランダム抽出の画像データを使用した場合と比較すると、人間の目で判断できる画像で学習を行うことで判定精度が向上した。これらの結果から、映画はコンテンツ全体から表現される感性情報に対し、必ずしも全てのシーンで同一の感性を表現している訳ではないため、画像をベースに学習を行う場合は、適切なシーンを選択する必要があることが分かった。

画像データセットを整備したら、データの学習させる回数（epoch）を大きく調整すると、結果として、90%以上までに学習精度を向上させた。

図 3-6 は機械学習の精度の変化を示す。

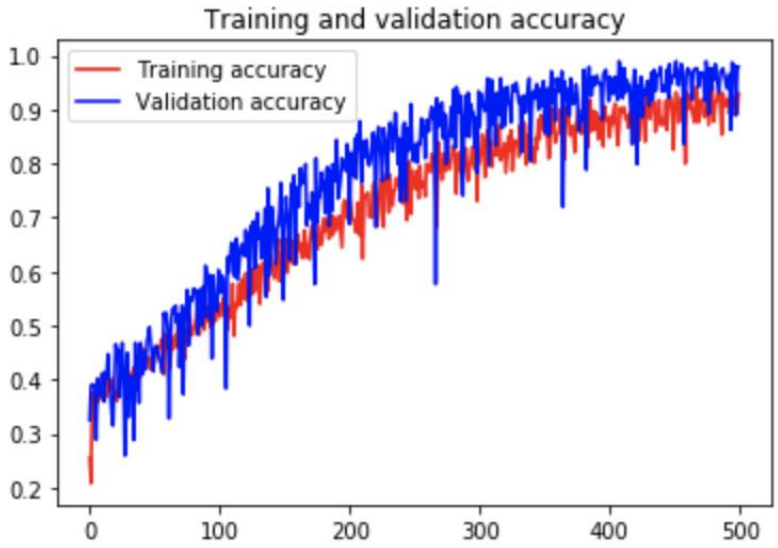
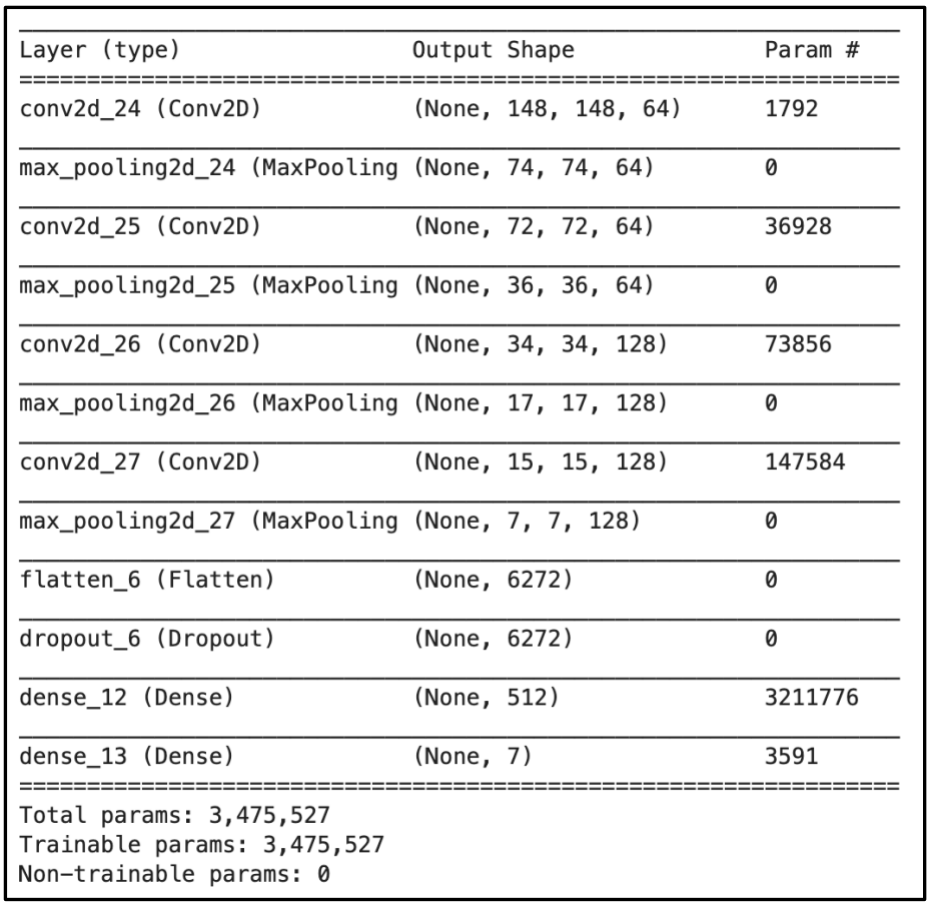


図 3-6　学習精度の変化

Fig.3-6 Machine leaning accuracy’s change

3.6 機械学習モデルの保存と映画感性データの計算



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_24 (Conv2D)           (None, 148, 148, 64)      1792
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 74, 74, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 72, 72, 64)        36928
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 34, 34, 128)       73856
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 17, 17, 128)       0
_________________________________________________________________
conv2d_27 (Conv2D)           (None, 15, 15, 128)       147584
_________________________________________________________________
max_pooling2d_27 (MaxPooling (None, 7, 7, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 6272)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 6272)              0
_________________________________________________________________
dense_12 (Dense)             (None, 512)               3211776
_________________________________________________________________
dense_13 (Dense)             (None, 7)                 3591
=================================================================
Total params: 3,475,527
Trainable params: 3,475,527
Non-trainable params: 0
```

図 3-7 訓練済み学習モデル

Fig.3-7 The deep learning model that finished learning

Keras の Model. predict ()を使い、上述の保存した学習済みの機械学習モデルで、新しい画像の感性を予測することができる。

```
# model.predict()のコード
predictions = model.predict(x)
print(predictions)
```

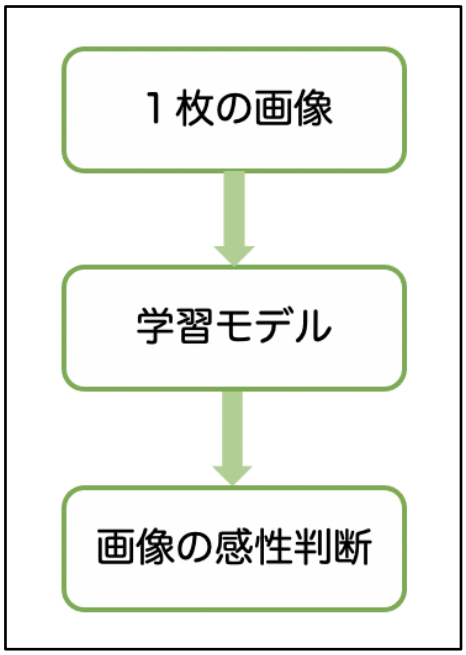


図 3-8 学習モデルを利用し、一枚の画像の感性を判断する

Fig.3-8 Using the learning model to predict the sensitivity of a single image

結果は以下(図 3-9)に示す。「楽しさ、怒り、安心、嫌悪、興奮、恐怖、悲しみ」7 感情分類のそれぞれの値を示している。

```
[[0.31179103 0.11470152 0.11470152 0.11470152 0.11470152 0.11470152
  0.11470152]]
```

図 3-9 学習モデルでの予測結果

Fig.3-9 Predictions using deep learning model

図 3-9 の通り、「楽しさ、怒り、安心、嫌悪、興奮、恐怖、悲しみ」の中で、一つ目の値が一番大きいので、この画像を「楽しみ（amusement）」と判断する。

そして、映画の予告編ビデオから、自動的に複数の画像を抽出し、保存するプログラムを作った。

このプログラムは、まず、サーバ上に保存した映画予告編ビデオを読み込み、そのビデオから抽出する画像を保存するための新しいフォルダーを作る。そして、自動的に予告編ビデオから、複数の画像を抽出した上で、該当する名前のフォルダーに保存する。

映画感性を判断する際に、先に各フォルダーの画像を読み込む。そして、その一つの映画予告編から抽出した画像を全て機械学習済みのモデルにより感性分類を行い、各感性の画像の割合を計算して、この映画の感性構成を推測した。一つの映画の感性データを得る過程は図 3-10 に示す。

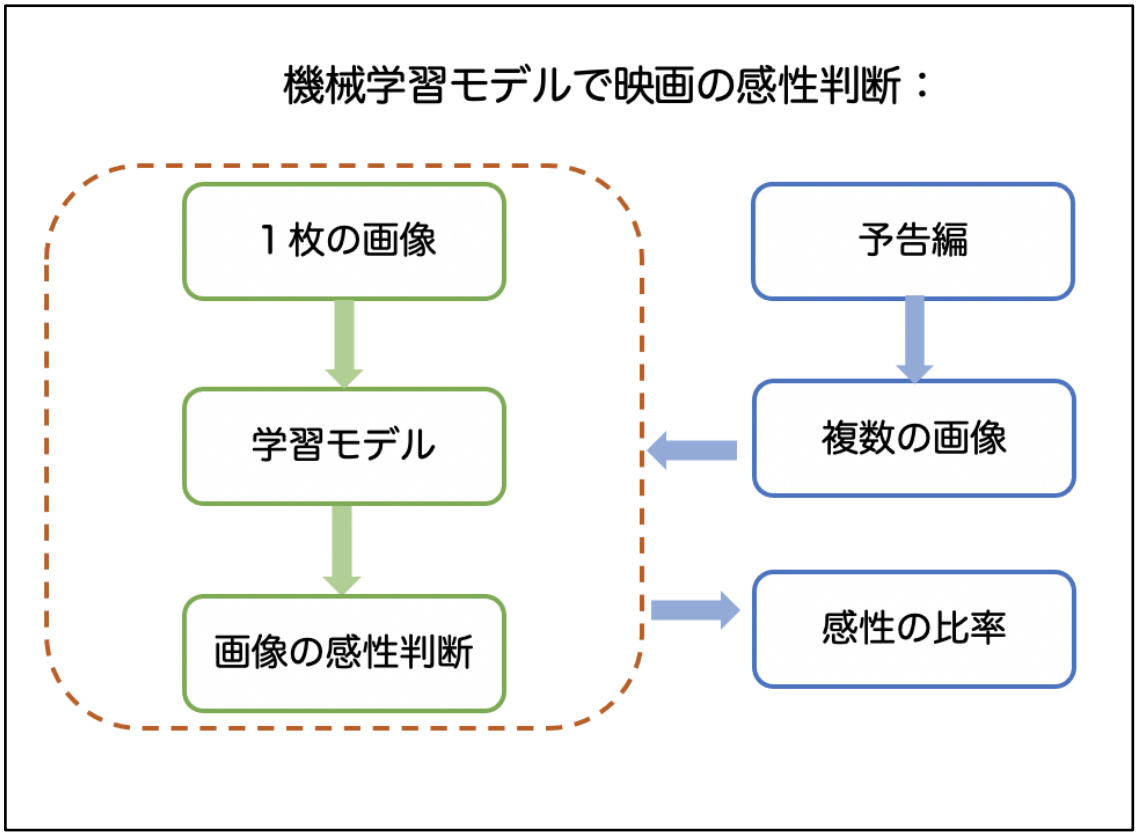


図 3-10 学習モデルでの予測結果

Fig.3-10 Predictions using deep learning model

このように、一つの映画の 7 感情の各感情の割合を得られる。

そして、得た感性データ（図 3-11）を映画の感性データベースに記録する。

```
amusement: 44.83%
anger:0.00%
contentment:13.79%
excitement: 0.00%
disgust:17.24%
fear:24.14%
sadness:0.00%
```

図 3-11 一つの映画の感性データ

Fig.3-11 Emotion data of one movie

大量の映画の感性データをこのように計算し、記録することによって、映画推薦システムためのデータベースを構築することができる。

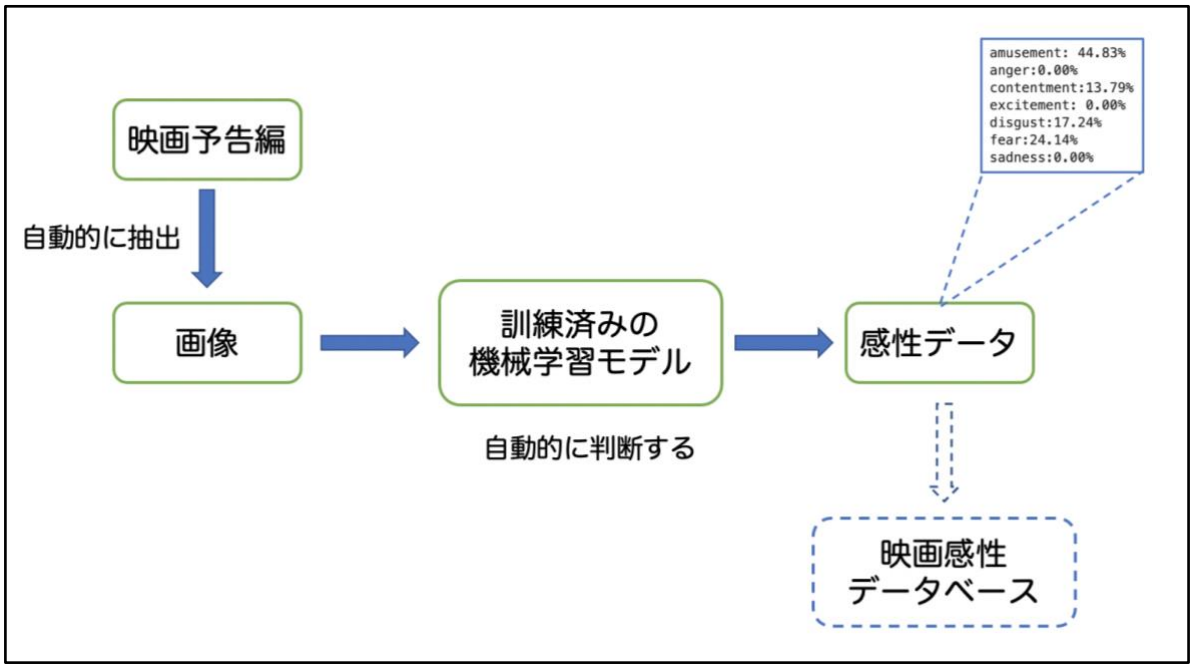


図 3-12 映画感性データを得る全体的過程

Fig.3-12 Overall process of obtaining movie sensitivity data

映画感性データの得る全体的過程は図 3-12 に示す：

サーバに保存した映画予告編ビデオを自動的に抽出してから、自動的に抽出した複数枚の画像をサーバに保存する。また、訓練済みの機械学習モデルを利用し、保存した一枚ずつの画像の感性を判断してもらう。その上、その一つの映画の各感情に判断された画像の割合を計算し、その映画の感性データとしてデータベースに記録する。

3.7 実験結果と結論

機械学習で、映画の画像を感性によって分類する実験は 90%以上の学習精度に達成した。

機械学習で、映画から抽出した画像を感性で分類することにより、映画を感性による分類を行うことができるとわかった。

この訓練済み学習モデルを利用し、映画推薦システムを構築するためのデータベースを作ることが可能である。

第 4 章：映画推薦システムの開発

4.1 概要

本章では、前述の機械学習を用いた、映画推薦システム Web アプリケーションの開発について説明する。

本研究は、Flask という Python 用ウェブアプロケーションフレームワークと HTML を使い、ウェブアプリケーションの開発を行った。

4.2 使用したツール

プログラミングの開発言語は主に Python である。Flask という Python 用のフレームワークを使い、ウェブアプリケーションを構築した。フロントエンドは HTML と CSS を使い、開発した。データは OpenPyXL という Python 用のライブラリを使い、Excel ファイルに記録したり、読み込んで、検索したりする。

（１）Flask（フラスク）について

Flask（フラスク）はプログラミング言語 Python 用の、軽量なウェブアプロケーションフレームワークである。

Flask は軽くて、必要最低限の機能しか持っていないが、拡張可能なフレームワークとして設計されているので、必要な拡張パッケージを加えれば、機能の実現ができる。

Flask には 3 つの主な依存関係がある。ルーティング、デバッグ、および Web サーバーゲートウェイインターフェイス(Web Server Gateway Interface， WSGI)。サブシステムは、Werkzeug から提供されている。テンプレートのサポートは Jinja2 によって提供されている。コマンドライン統合は Click から行われる。

4.3 推薦システムの開発

第 2 章に説明したように、本システムは主に 2 つサブシステムに構成されている。

1 つ目は、映画感性データベースの構築である。この部分は、第 3 章で紹介した機械学習モデルを使い、自動的に映画の 7 感情の比率を記録し、映画感性データベースを作った。

2 つ目のサブシステムは、感性に基づく映画を推薦する機能である。この部分は、まず映画とユーザ要求のマッチングのアルゴリズムを比較し、ピアソンそう関係数を用いて、映画感性データベースの中からユーザ要求と近い映画を抽出し、推薦を行う。

以下にシステムの開発についての詳細を説明する。

4.3.1 映画感性データベースの構築

映画感性データベースには、レコメンデーションを行うために、多くの映画の感性情報を保存するべきである。この映画感性データベースを構築するために、まず、第 3 章で紹介した画像を感性で分類する機械学習の結果をモデルとして保存した。また、映画の予告編映像から自動的に画像を抽出するプログラムを実現した。最後に、予告ビデオから抽出された画像を自動的に機械学習モデルによる分類を行うプログラムを開発した。その画像ごとに感性を分類した後、映画ごとに、各感性の割合を計算し、データを記録して、データベースの構築ができた。

結果として、映画の予告編映像をサーバ上に保存し、自動的にその映画の 7 つの各感性の値をデータベースに保存した。

データの記録、読み込み、検索の実現は Python 用のライブラリ OpenPyXL を使って行う。映画感性データベースは図のように、Excel の形で構築された。

movie_name	amusement	anger	contentmen	disgust	excitement	fear	sadness
Le Grand Méchant Renard et autre	42.8571429	0	0	23.8095238	0	33.3333333	0
1917	0	6.25	0	0	0	43.75	50
A Quiet Place Part II	12.5	12.5	0	0	0	75	0
a rainy day in new york	15.3846154	7.69230769	2.56410256	10.2564103	0	53.8461538	10.2564103
Ad Astra	5.88235294	11.7647059	5.88235294	0	0	64.7058824	11.7647059
American Factory	45.9459459	32.4324324	0	5.40540541	0	5.40540541	10.8108108
April and the extraordinary world	42.3076923	30.7692308	7.69230769	7.69230769	0	3.84615385	7.69230769
Aquaman	0	6.66666667	0	0	0	93.3333333	0
Avengers	26.6666667	13.3333333	6.66666667	33.3333333	0	20	0
becky	44.8275862	0	13.7931034	17.2413793	0	24.137931	0
Before Midnight	0	11.1111111	7.40740741	7.40740741	0	74.0740741	0
Billy Lynn's Long Halftime Walk	35.2941176	11.7647059	5.88235294	5.88235294	0	41.1764706	0

図 4-1　映画感性データベース-Movie Emotion シート

Fig.4-1 Emotion database Movie Emotion sheet

機械学習のモデルを用い、各感性の値を計算することができ、映像があれば、映画の感性データを随時に更新することができる。そのため、新しい映画でも、予告編があればすぐ推薦することができる。

検証実験のため、データベースに 103 本の映画の感性情報を記録した。

映画の感性データ以外、ウェブアプリケーションを構築する、または検証・分析のため、データベースの Excel には他の内容も記録している。

表 4-1 データベースの構築

シート名	データ
MOVIE EMOTION	映画名、感性情報（７感情それぞれの割合）
RESULT	ユーザの要求、 要求に類似度が高い結果の映画名、ユーザ IP
FEEDBACK	返した映画名、ユーザ IP、フィードバック

4.3.2 推薦機能の開発

推薦機能は具体的に、ユーザが入力した感性要求と映画の感性とをマッチングし、合致している映画をユーザに返す機能である。

本研究の中で、ユーザの感性要求と映画の感性を 2 つのベクトルにする。ユーザに見たい映画の 7 つ感情を数値で入力してもらう。また、データベース中の映画の感性情報である、7 感情のデータもベクトルにする。

この 2 つのベクトルに対し、推薦アルゴリズムで類似度を計算し、比較する。映画感性データベースの中で、ユーザの要求に合致している映画をデータベースに記録し、ユーザに推薦する。

推薦アルゴリズムは、具体的にユークリッド距離、コサイン類似度とピアソン相関係数などがある。

また、各アルゴリズムの比較を説明する。

4.3.2.1 マッチングアルゴリズム

（1）ユークリッド距離

ユークリッド距離（Euclidean distance）またはユークリッド計量（Euclidean metric）とは、ユークリッド空間（Euclidean space）中、二つ点の間の距離である。ピタゴラスの公式（Pythagorean theorem、勾股弦の定理）によって与えられる。この公式を距離函数として用いればユークリッド空間は距離空間となる。

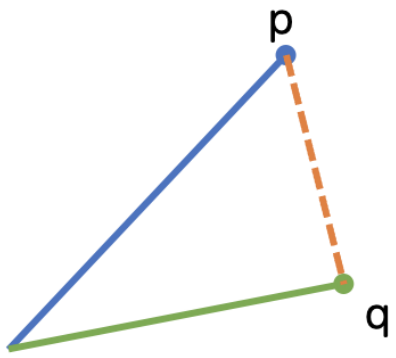


図 4-2　二次元ユークリッド距離の原理

Fig.4-2 Euclidean distance

図 4-2 通り、ユークリッド平面において、点 p=(p1, p2)と点 q=(q1, q1)の間の距離 ab をユークリッド距離で計算する。

一般的に、N 次元空間において、ユークリッド距離の公式は以下である：

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_i - q_i)^2 + \ldots + (p_n - q_n)^2}.$$

（２）コサイン類似度

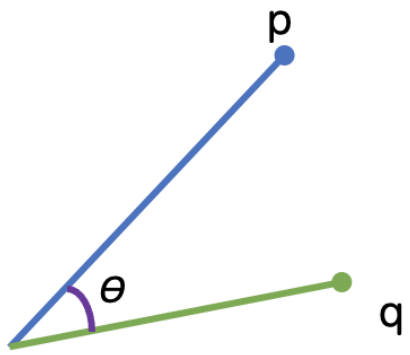


図 4-3　二次元コサイン類似度の原理

Fig.4-3 Cosine’s geometric principles

コサイン類似度は、下記の数式で表され、ベクトルのなす角のコサインを表す。

$$cos(\theta) = \frac{\sum_{k=1}^{n} x_{1k} x_{2k}}{\sqrt{\sum_{k=1}^{n} {x_{1k}}^2} \sqrt{\sum_{k=1}^{n} {x_{2k}}^2}}$$

図 4-3 で表しているように、2 つのベクトルの方向が近いほど、そのなす角は小さくなり、コサイン類似度は 1 に近くなる。なす角の向きが異なるほど大きくなり、コサイン類似度が-1 に近くなる。

（３）ピアソンの積率相関係数

ピアソンの積率相関係数（相関係数、Pearson correlation coefficient）とは二つの変数の間にある線形な相関の強弱を測る指標である。ピアソン相関係数の特性の一つは、二つ変量の位置や尺度の変化は係数に影響を与えないことである。つまり、X を a + bX に移動し、Y を c + dY に移動しても、2 つの変数の相関係数は変わらない。a, b, c, d は定数である。

ピアソン相関係数の公式は以下通り：

$$r = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{j=1}^{n} (x_j - \overline{x})^2} \sqrt{\sum_{k=1}^{n} (y_k - \overline{y})^2}}$$

公式から見ると、ピアソン相関係数はコサイン類似度と非常に類似している。ただ、ベクトルの各数値は先に平均値を引いてから計算することで、中心化を行う。つまりピアソン相関係数は中心化したコサイン類似度である。

4.3.2.2 マッチングアルゴリズムの比較

上述の三つのアルゴリズムを比較する。

ユークリッド距離は数値の大きさに大きく影響される。コサイン類似度とピアソン相関係数はほぼ同じく、数値の大きさではなく、ベクトルの傾向をよく反映する。

例えば、data1 = [1, 2, 3]、data2 = [0, 1, 3]、data3 = [2, 4, 6]、data4 = [3, 6, 9]。三つのアルゴリズムで計算した結果を正規化し、表 4-2 に整理した:

表 4-2 マッチングアルゴリズムの比較

	ユークリッド距離	コサイン類似度	ピアソン相関係数
data1 と data2	0.41	0.96	0.99
data1 と data3	0.21	1	1
data1 と data4	0.12	1	1

結果から見ると、数値の大きさはユークリッド距離の結果を大きく影響している。

その一方、コサイン類似度とピアソン相関係数は数値の大きさに影響されずに、データの傾向を反映している。

本研究の映画感性データベースは、ある映画で、各感情の比率を記録したデータベースのため、各感情の激しさ（程度）を反映していない。また、映画推薦システムを利用する場合、ユーザ一人ひとりの感情に対する尺度が異なるため、入力した数値の大きさではなく、各感情のデータの間の関係性に注目すべきだと考える。

そのため、本システムはピアソン相関係数を利用し、ユーザの要求と映画の感性データのマッチングを行った。

4.3.2.3 推薦機能の実現

本研究の中で、具体的に 2 つのベクトルは、一つ目はユーザの要求、つまり入力してくれる 7 感情の要求の値である。もう一つはデータベース中の映画の感性情報、7 感情のデータである。この 2 つのベクトルのピアソン相関係数を計算し、また結果を正規化（normalization）して、値を(0, 1)にする。その上で、値を比較し、結果が大きい方を推薦する。

例えば、ユーザが[60, 0, 30, 0, 0, 0, 10]を入力した場合、このデータをベクトルに変換して、データベース中の映画情報から変換したベクトルとコサイン類似度の計算を行う。図 4-4 に示しているように、「The Vanished」という映画の感性データは[25.00, 25.00, 6.25, 9.38, 0.00, 31.25, 3.31]であり、「Get Out」という映画の感性データは[16.22, 18.92, 0, 0, 2.70, 62.16, 0]である。

movie_name	amuser	anger	conten	disgust	exciten	fear	sadness
The Vanished	25.00	25.00	6.25	9.38	0.00	31.25	3.13
Get Out	16.22	18.92	0	0	2.70	62.16	0

図 4-4　映画「The Vanished」の感性データ

Fig.4-4 Emotion data of movie *The Vanished*

この二つのベクトルの類似度を計算した結果は：ピアソン相関係数は 0.58 と 0.43 である。

つまり、結果から見ると、この二つの映画から、「The Vanished」をユーザに推薦すべきである。

プログラミングは付録 2.を参照すること。

全てのピアソン相関係数を計算した後、値の高い順で、1～５番目までの映画名を、Excel に記録し（図 4-5）、ユーザに返す。

夜明け告げるルーのう	Valkyrie	No Time to Die	天気の子	American Factory
Knives Out	英雄本色	名探偵コナン 紺青の	La La Land	Shazam!
天気の子	Ryuichi SakamotoCODA	Song of the Sea	Doctor Sleep	sherlock
Inglourious Basterd	Boarding School	Irresistible	The Painter and	Valkyrie
Irresistible	The Painter and The	The Rental	The Rental	Boarding School
夜明け告げるルーのう	天気の子	Valkyrie	No Time to Die	The Rental
天気の子	夜明け告げるルーのうた	The Rental	death_on_the_ni	Kusama - Infinity
Irresistible	Spontaneous	Boarding School	Boarding School	天気の子
Boarding School	Irresistible	The Painter and T	Valkyrie	Inglourious Basterds
The Painter and The	Valkyrie	夜明け告げるルーの	天気の子	The Rental

図 4-5　映画感性データベース-Result シート

Fig.4-5 Emotion database Result sheet

4.4 ウェブアプリケーションの開発

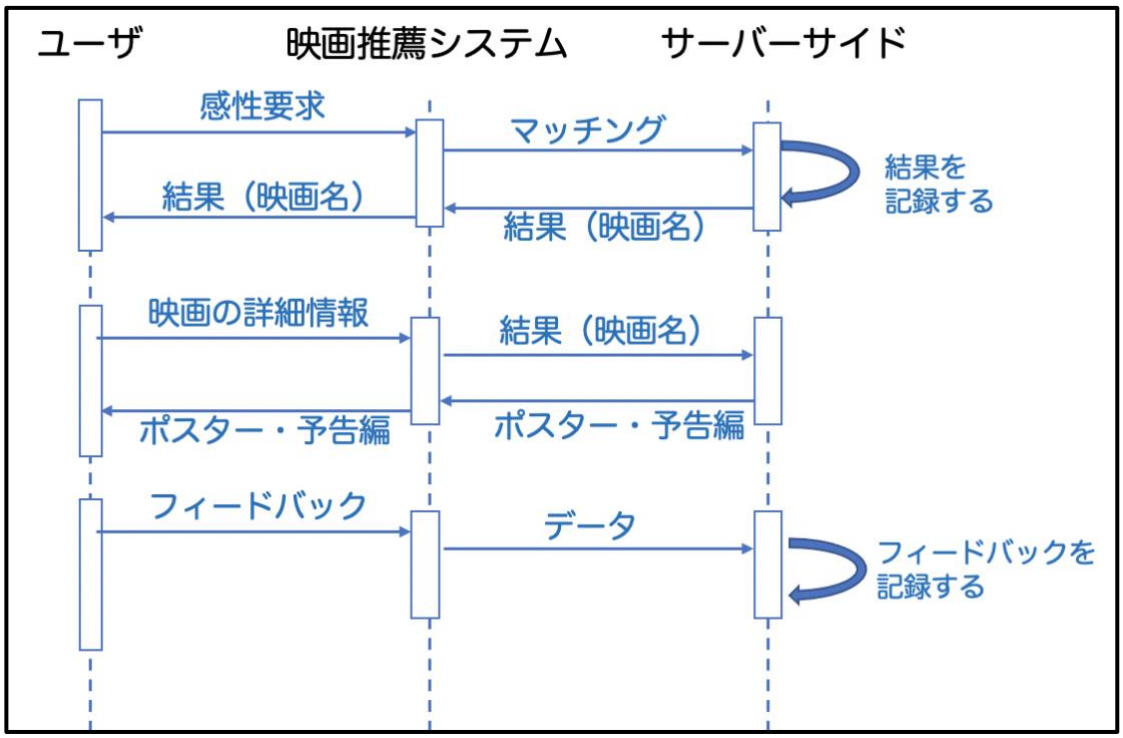


図 4-6 システムシーケンス図

Fig.4-6 system design diagram

（1）映画推薦システムの詳細設計

システムシーケンス図は図 4-6 に示す。

映画推薦システムはユーザから感性要求をもらい、サーバーサイドでデータベース中の映画とのマッチングを行う。マッチングの結果をデータベースに記録する。

そして、データーベースから、結果を読み込み、ウェブアプリケーションを通じて、ユーザに返す。

ユーザが映画の詳細情報を見る要求を送る際に、推薦システムはサーバサイドに推薦した結果の映画名をサーバサイドに渡し、その映画のポスターや予告編を推薦アプリケーションに渡し、ユーザに提示する。

また、ユーザがフィードバックを行う際に、その結果の値をサーバーサイドに渡し、データベースに記録する。

（2）ウェブアプリケーションの構成

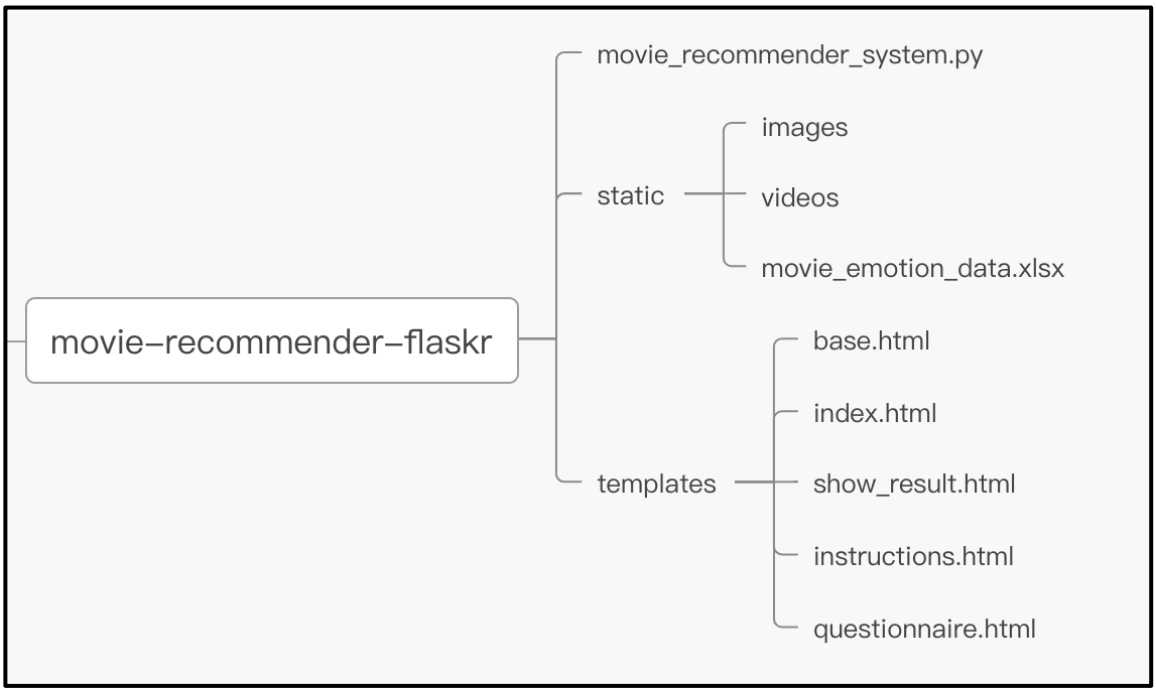


図 4-7 ウェブアプリケーションの主要構成図

Fig.4-7 Web application's architecture

本研究は、Flask という Python 用ウェブアプロケーションフレームワークと HTML を使い、ウェブアプリケーションの開発を行った。ウェブアプリケーションの主要構造は図 4-7 が示している通りである。

movie_recommender_system.py は主要な動作するプログラムである。

「static」にウェブページに展示する画像を保存し、また映画の詳細情報のポスターや予告編なども保存している。また、データベースの movie_emotion_data.xlsx も配置している。

「templates」フォルダには HTML ファイルを配置する。

具体的に、このウェブアプリケーションを使う際に：

index.html（/）のページで、ユーザが感性要求を入力し、Flask は、form からデータをサーバーサイドに値を渡す。

そのユーザの要求とデータベース中の映画感性のコサイン類似度の計算を行い、類似度の高い映画の映画名を Result という Excel のシートに記録する。

ユーザは show_result.html（/show）にアクセスする際に、その IP アドレスの一番新しい記録した結果をフロントエンドに返す。

また、その映画の予告編やポスターなども提示する。

もしユーザがフィードバック機能を利用し、フィードバックをすれば、Flask は form から値をサーバーサイドに渡し、データベースの Excel の Feedback というシートに記録する。

図 4-8 のように、一番新しい推薦結果と、I P アドレス、フィードバックの結果（ユーザが入力した感情と合致していると思う場合は Y、合致していないと思う場合は N を記録する。）をデータベースの Feedback シートに記録している。

Soul	175.134.206.183	N
The Rental	175.134.206.183	Y
Hacksaw Ridge	175.134.206.183	Y
天気の子	175.134.206.183	
1917	175.134.206.183	Y
コンフィデンスマンJP プリンセス	175.134.206.183	
Spontaneous	60.150.241.223	
Irresistible	60.150.241.223	

図 4-8 映画感性データベース-Feedback シート

Fig.4-8 Emotion database Feedback sheet

4.5 完成した映画推薦システム

完成した感性情報に基づく映画推薦システムのイメージを示す。

まず、図 4-9 のように、メニューバーには 3 つメニューがあり、それぞれは「Movie Recommender」、「Instructions」と「Questionnaire」である。それぞれは、「/」、「/instructions」と「/questionnaire」につながっている。

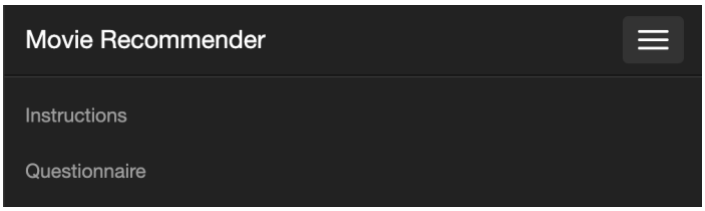


図 4-9 システムのメニューバー

Fig.4-9Menu bar of the system

「Movie Recommender」(/recommender) のページは図 4-10 のイメージ通り、ユーザが感性要求の数値を入れるフォームがあり、一番下には推薦する結果を示している。

図 4-10 映画感性データベース-Feedback シート

Fig.4-10 Emotion database Feedback sheet

その結果をクリックすれば、図 4-11「/show」につながり、推薦する映画のポスターと予告編を展示するページである。

またフィードバックのフォームと他の推薦結果がある。

図 4-11 推薦結果のページ

Fig.4-11 Recommendation result web page

図 4-12 のように、「Instructions」のページにシステムの使用説明を提示し、「Movie Recommender」（/recommender）につながるボタンを設置している。

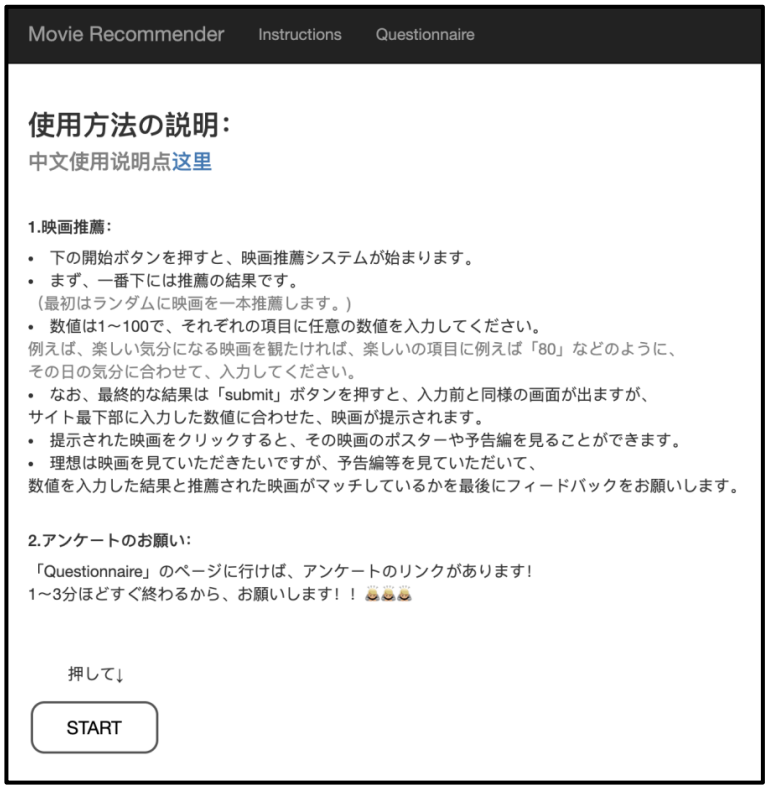


図 4-12 「Instructions」ページ

Fig.4-12 Instruction web page

図 4-13 のように、「Questionnaire」(/questionnaire)のページには、Qualtrics 上で公開されたアンケートへのリンクと Q R コードを示している。

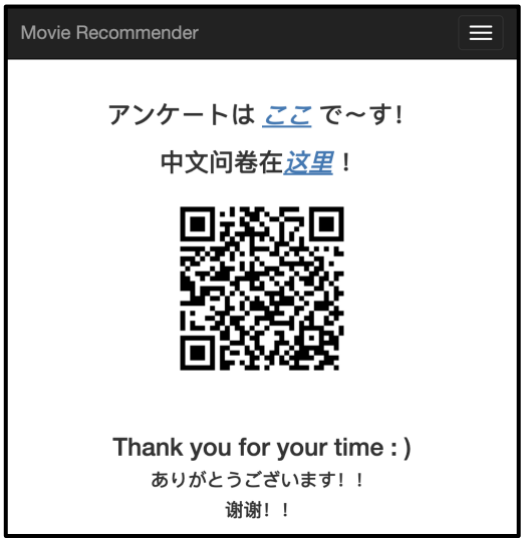


図 4-13 「Questionnaire」ページ

Fig.4-13 Questionnaire web page

第５章：検証実験

5.1 概要

本章では、提案及び開発した感性情報に基づく映画推薦システムに対して行った検証実験について説明する。

また、検証実験の結果から、本研究で提案及び開発した映画推薦システムの有効性を検証する。

5.2 検証実験の目的

検証実験の目的は、複数の人に使ってもらった上で、感性情報に基づく機械学習を用いた映画推薦システムに関する評価と感想を獲得し、映画推薦システムに対する一番重要な「感性に基づく映画推薦する」機能の有効性を検証することにより、システムの有効性を検証することである。

5.3 検証実験の内容

「感性に基づく映画推薦する」機能の有効性を検証するために、推薦の正確さ、または、推薦した結果に対するユーザの見る意欲、システムを使う意欲など重要な項目を検証すべきだと考える。

推薦の正確さは非常に重要な項目と考え、より有効に検証できるために、システムの結果のページにフィードバックの機能を加えた。毎回の推薦結果が要求に合致しているかのデータを収集することができる。また、このデータを利用し、推薦の調整、機械学習の調整などで、システムの推薦の正確さを向上させることが期待できる。

フィードバックの機能を利用し、検証実験を 2 つ部分に分けて設計した。

1 つ目は本研究で開発した感性情報に基づく映画推薦システムのウェブアプリケーションを被験者に体験してもらい、推薦の正確さを数回もフィードバックしてもらった。

2 つ目はシステムを全体的に体験した上で、アンケートに回答することである。

また、複数の被験者にインタビューを行った。

具体的な過程は以下通り：
まず、被験者に本映画推薦システムの使い方について説明を行う。

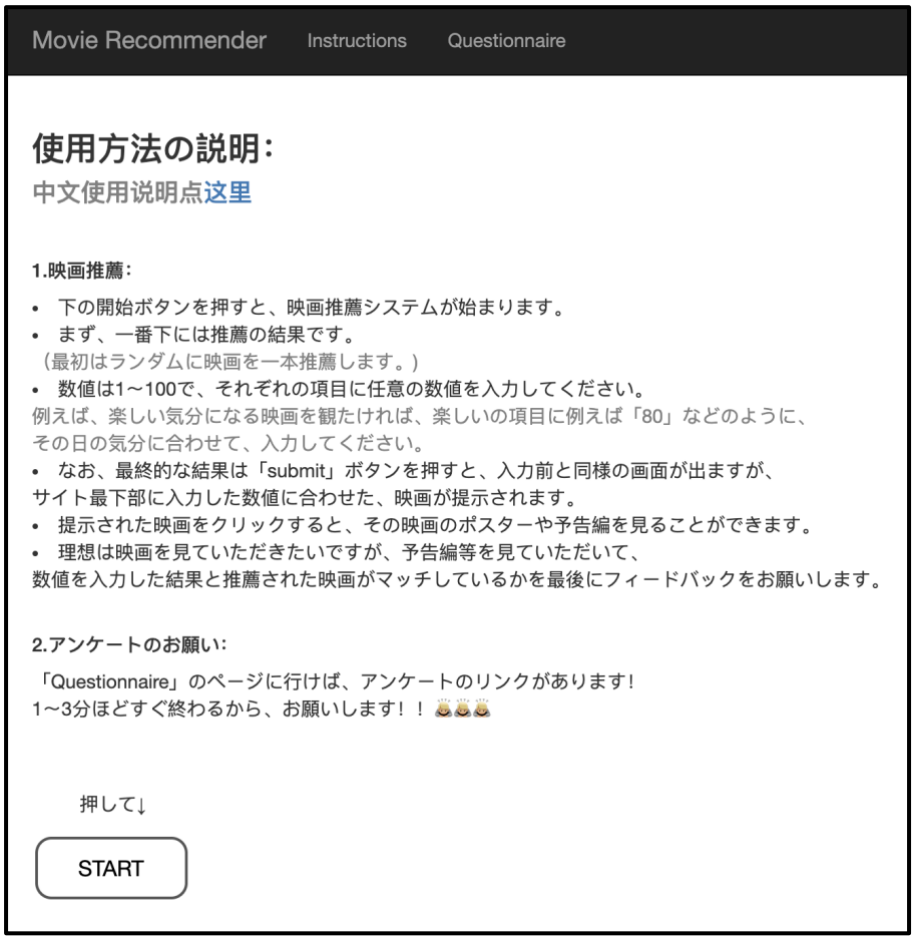


図 5-1 使用方法説明ページ

Fig.5-1 Emotion database Result sheet

また、推薦機能を複数回体験してもらい、フィードバックを送信してもらった。

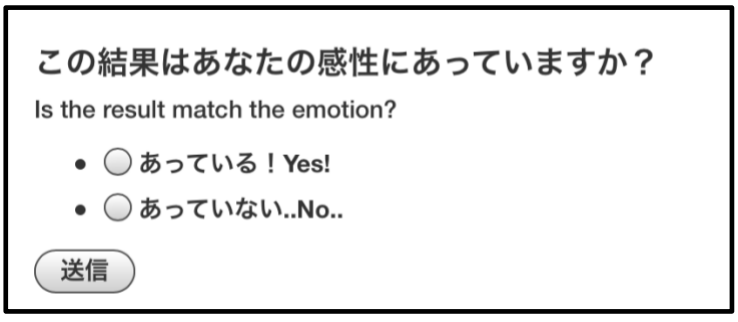


図 5-2 フィードバック form

Fig.5-2 Feedback form

体験が終わった後、推薦システム全体に関するアンケートを行う。
アンケートの内容は「付録 1.機械学習を用いた感性情報に基づく映画推薦システムに関するアンケート」を参照すること。

5.4 検証実験の実施

検証実験は 2 つ部分があり、オンラインで実施した。
1 つ目は、被験者に自由にシステムを体験してもらい、複数の推薦結果の精度についてフィードバックを行なってもらえた。
2 つ目は Qualtrics 上で公開されたアンケートに答えてもらった。
アンケートの詳細は、「付録 1.機械学習を用いた感性情報に基づく映画推薦システムに関するアンケート」を参照すること。
検証実験は 2021 年 1 月 9 日から始まり、被験者は合計 27 人である。
表のように検証実験の詳細を記載する。

表 5-1 検証実験

実施日程	2021 年 1 月 9 日～
実施対象	SDM の関係者、他の知り合いなど
実施環境	オンライン
実施方法	アンケート
回答数	27

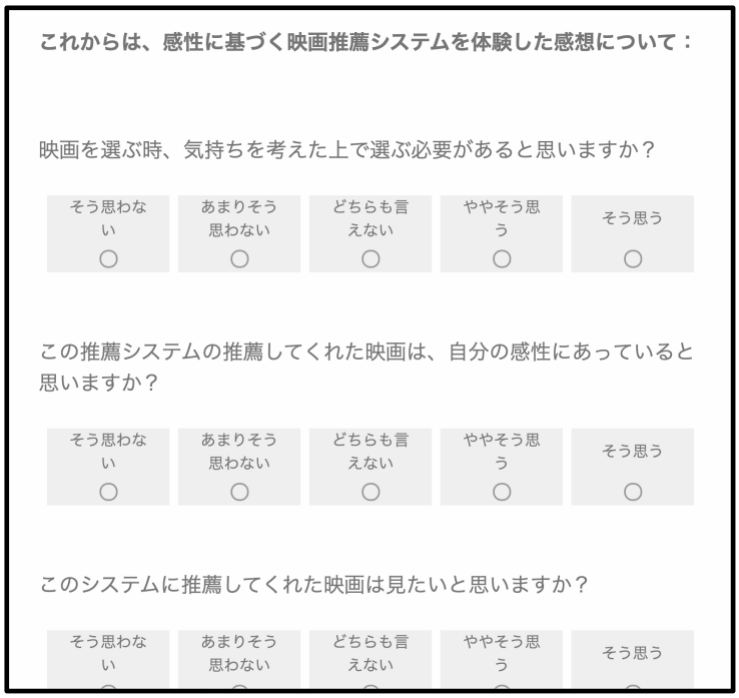


図 5-3 公開されたアンケートの様子

Fig.5-3 Questionnaire

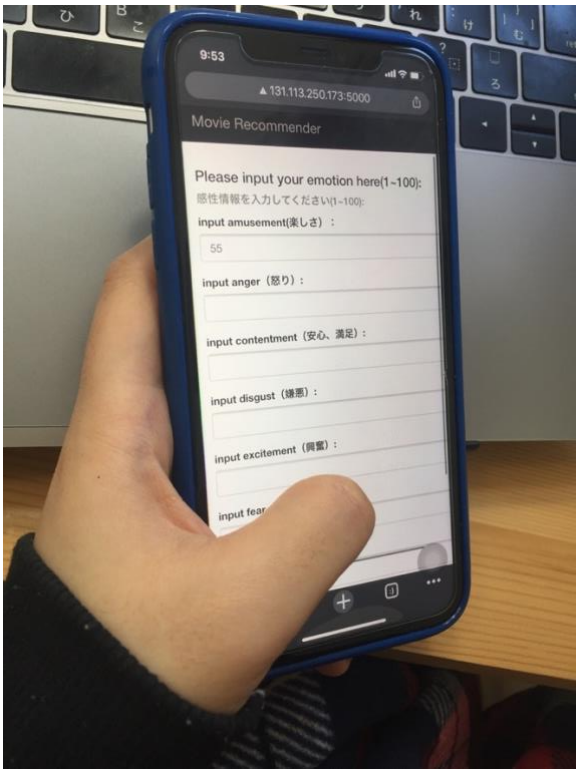


図 5-4 被験者にシステムを体験させる様子

Fig.5-4 A user is using the system

5.5 検証実験の結果と有効性分析

検証実験の結果を記録して、その結果に対しての分析を説明する。

まず、システム内に行なったフィードバックのデータは、回答が合計 56 件記録し、その中 38 件は推薦した結果があっていると答えた。約 68%である。

以下はアンケート調査の結果を説明する。

（1）被験者の属性

まずは、アンケートの回答者合計 27 名の属性は以下に示す。

年代別は、以下の図のように、20 代が一番多く 18 名、30 代が 3 名、40 代 2 名、50 代 3 名、60 代 1 名である（図 5-5）。

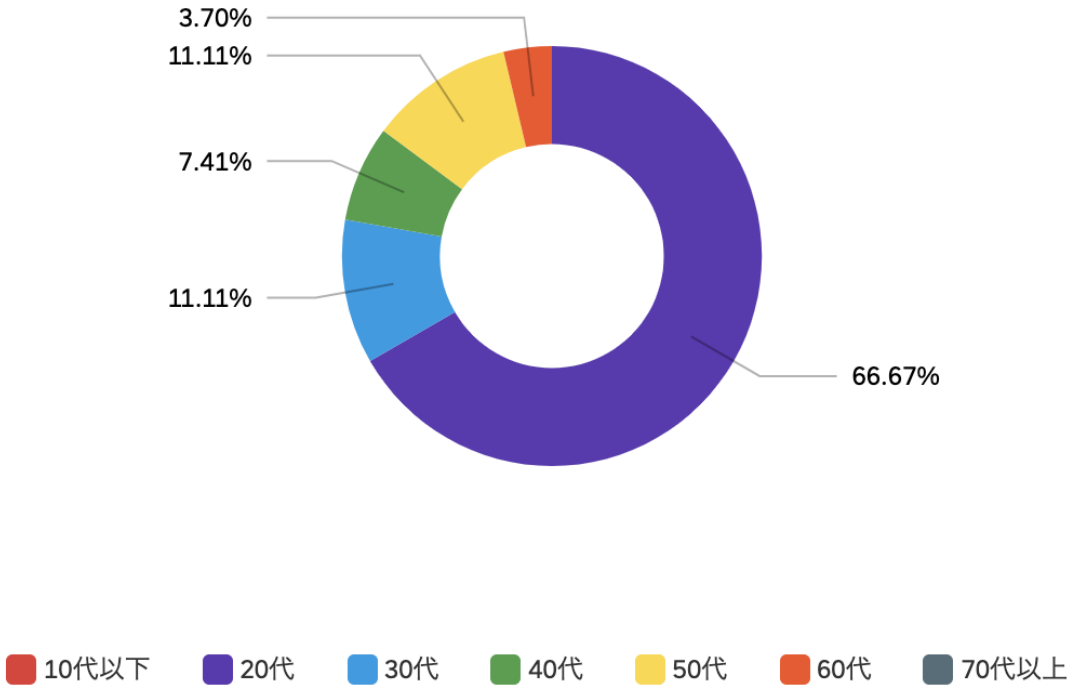


図 5-5 被験者年齢

Fig.5-5 age

性別では、女性が 18 名、男性が 9 名である。（図 5−6）

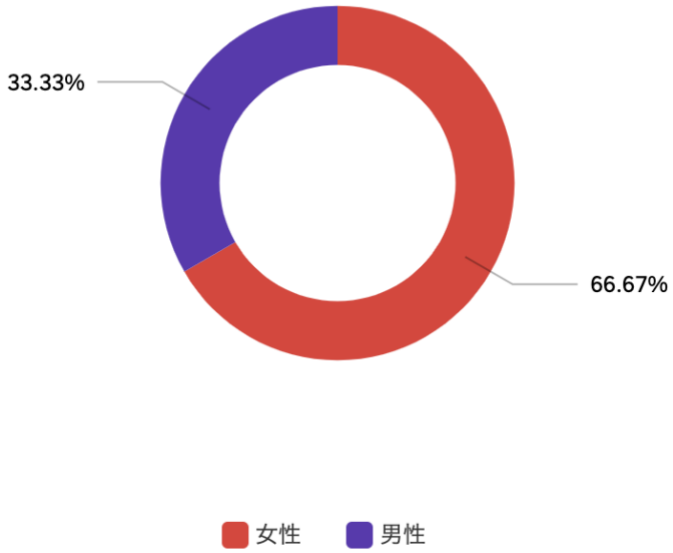


図 5-6 被験者性別

Fig.5-6 gender

映画を見る頻度については、平均 1 ヶ月に 1 回〜2 回が、8 名だった。あまり見ないのは 10 名、平均 1 ヶ月に 3 回〜5 回は 4 名、平均 1 ヶ月 6 回以上は 5 名である。（図 5-7）

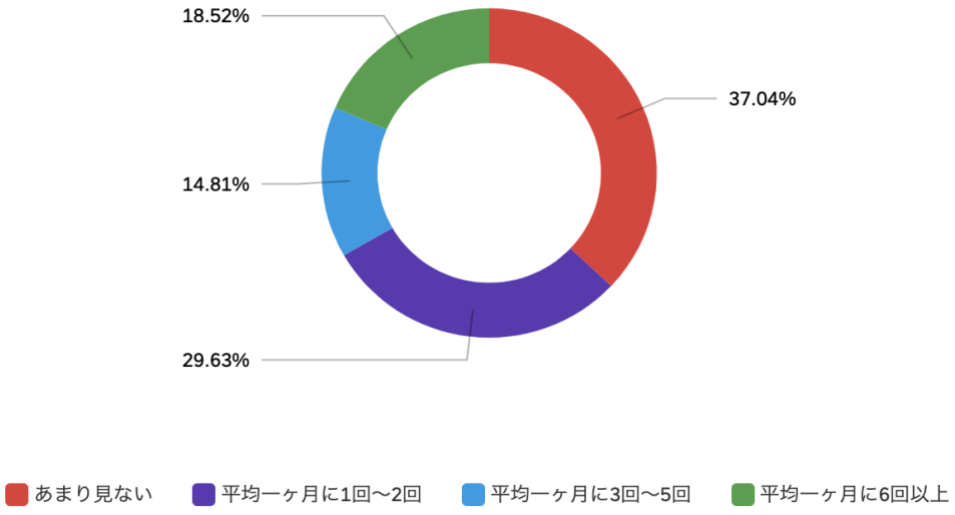


図 5-7 映画を見る頻度

Fig.5-7 Frequency of watching movies

「あなたは映画を見るとき、よく使っているサービスを教えてください（複数回答可）」という質問に対し、多くの被験者が動画配信サービス（Netflix、Amazon Prime Video、Hulu）を選択し、21 名だった。

映画館を選んだのは 15 名、家庭用 DVD・ブルーレイは 2 名、テレビは 3 名である（図 5-8）。

動画配信サービスを利用する人が 21 名で、割合は 78%に達した。

図 5-8 映画を見る手段

Fig.5-8 Method of watching movies

（2）本システムの利用体験について

「映画を選ぶとき、気持ちを考えた上で選ぶ必要があると思いますか？」という質問に対し、「ややそう思う」と「そう思う」と回答した人は 85%で、この質問に対して否定の意見を持つ回答は 4%しかないことが明らかになった（図 5-9）。

つまり、多くの人にとって、映画を選ぶ際に、気持ちなど感性情報を考えることが必要だとわかった。

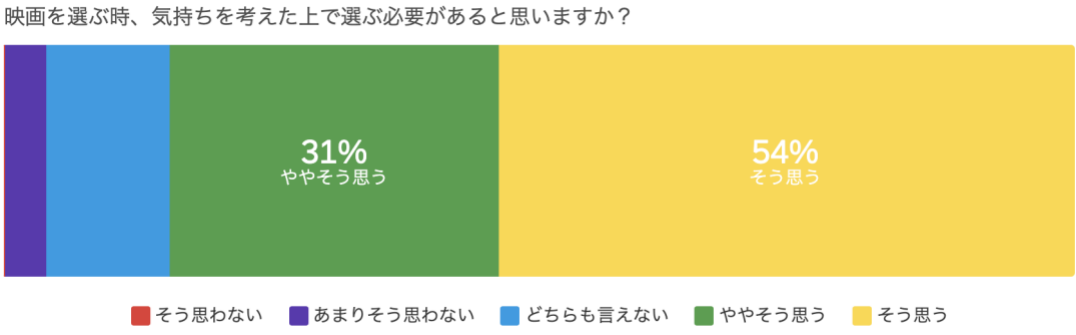


図 5-9 映画を選ぶとき、気持ちを重視しているか

Fig.5-9 Whether regard emotion important while choosing movies

「この推薦システムの推薦してくれた映画は、自分の感性にあっていると思いますか？」という質問に対し、69%の被験者が賛成の意見を持ち、27%の被験者が「どちらも言えない」を選んで、否定する意見は 4%だった（図 5-10）。

図 5-10 推薦の正確さ

Fig5-10 Accuracy of recommender

「推薦してくれた映画は見たいですか？」という質問に対して、77%の被験者は賛成の意見を持っていて、16%の被験者は否定の意見を持っている。（図 5-11）

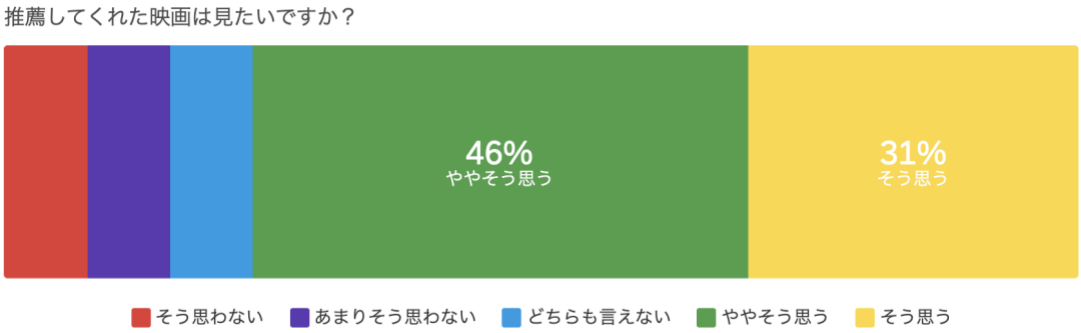


図 5-11 推薦した映画の見る意欲

Fig.5-11 How people what to watch the movie

「今後この映画推薦システムを利用したいと思いますか？」の質問に対し、96%の被験者が賛成の意見を持っている。否定の意見を持っているのは 4%しかいない（図 5-12）。

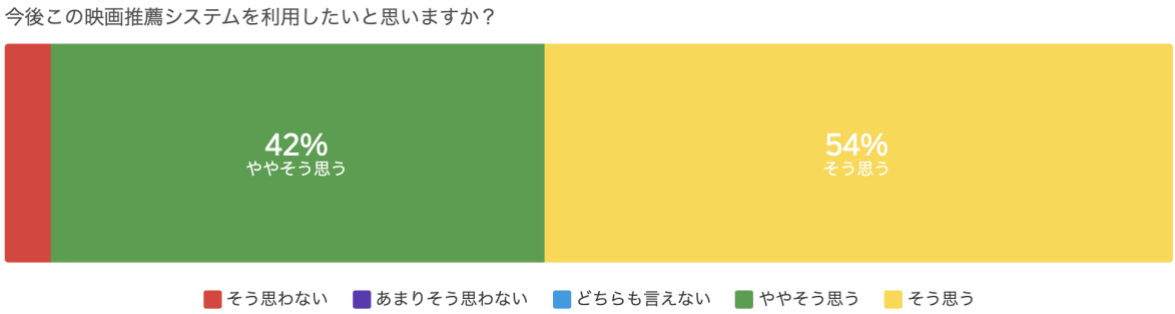


図 5-12 システムを利用する意欲

Fig.5-12 How people want to use this system again

検証実験の結果は以上であり、「感性による映画推薦する」機能とシステム全体的の有効性についての分析を説明する。

まず、「感性による映画推薦する」機能の有効性について、システム内で行なったフィードバックの結果は、正確に推薦していると思うのが約 68%である。アンケートの「この推薦システムの推薦してくれた映画は、自分の感性にあっていると思いますか？」という質問の結果：賛成しているのは 69%である。この 2 つのデータから、推薦する正確さはポジティブな評価をもらったが、より感性に合致するために、推薦方法や機械学習などの調整が必要だと考えられる。

また、システム全体の有効性について、アンケートからのデータを整理した。
本システムに対する体験に関する問題は、３つがある：

- この推薦システムの推薦してくれた映画は自分の感性に合っていると思いますか？
- このシステムに推薦してくれた映画は見たいと思いますか？
- 今後この映画推薦システムを利用したいですか？

また、この３つの問題の選択肢は５つに設計した：

- そう思わない
- あまりそう思わない
- どちらとも言えない
- ややそう思う
- そう思う

「どちらとも言えない」を「０」にして、システムの評価結果を数値化した。図 5-13 は各項目の平均値を示している。

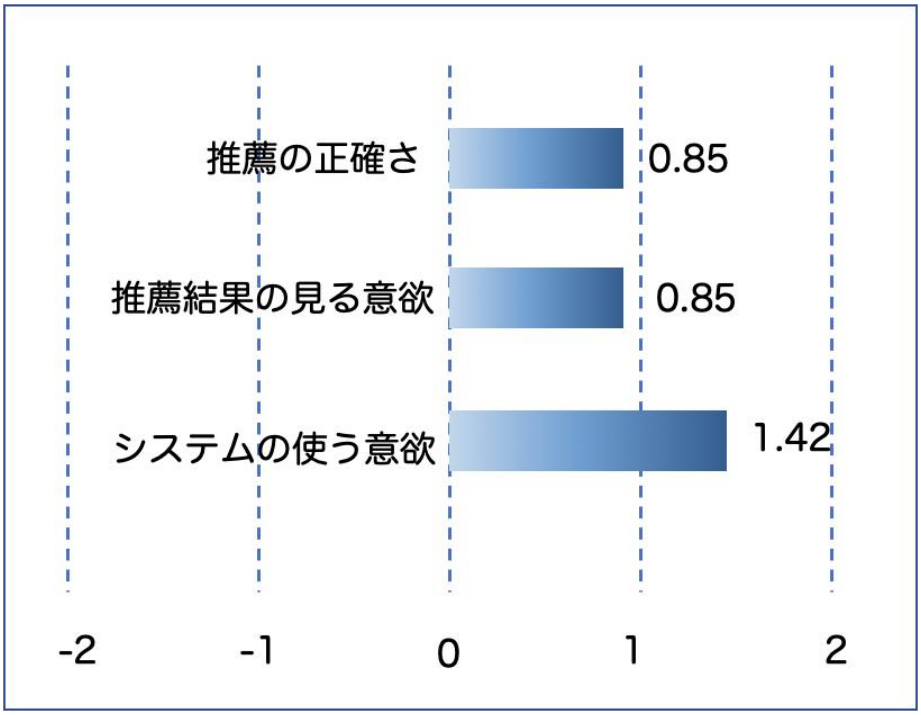


図 5-13 検証実験の平均値

Fig.5-13 Average of the result

システムの正確さ、推薦した映画を見る意欲、またシステムを利用する意欲はそれぞれ正の結果となり、積極的な結果だと言えるだろう。

映画推薦システムについて、システムの正確さ、推薦した映画を見る意欲、またシステムを利用する意欲から、本システムの主要な機能、「感性による映画推薦」の有効性が検証されたと考えられる。

5.6 アンケート調査のデータ分析

（1）仮説を立つ

アンケートの質問を整理すると、「年代、性別、映画を見る頻度、映画を見る手段、映画を選ぶとき気持ちに対する重視度、推薦の正確さ、映画を見る意欲とシステムを使う意欲」8 つの項目がある。

その 8 つの項目の間の関係を考えながら、以下の仮説を立てた：

- 映画を見る頻度と映画を選ぶ時の気持ちに対する重視度は正の相関がある。
- 映画を見る手段は本システムを使う意欲と相関がある。
- 推薦の正確さと映画を見る意欲は正の相関がある。
- 推薦の正確さとシステムを使用する意欲とは正の相関がある。
- 気持ちの重視度は性別と相関がある。
- 映画を見る手段は年代と相関がある。

（2）データ分析

アンケートの結果を整理した上で、相関分析を行った。

表 5-2 相関分析の結果

	年代	性別	映画を見る頻度	動画配信使用状況	気持ち重視度	推薦正確さ	見る意欲	使用する意欲
年代	1							
性別	0.40	1						
映画を見る頻度	-0.34	-0.13	1					
動画配信使用状況	-0.25	-0.18	0.38	1				
気持ち重視度	-0.30	-0.30	0.49	0.12	1			
推薦正確さ	-0.24	0.15	0.29	-0.11	0.20	1		
見る意欲	-0.21	0.10	0.44	-0.15	0.41	**0.79**	1	
使用する意欲	-0.43	-0.08	0.35	-0.05	0.18	0.64	0.66	1

アンケートの「あなたは映画を見るとき、よく使っているサービスを教えてください（複数回答可）」という質問の回答は、非常に多くの人は動画配信サービスを選んだ。またデータ処理の簡単さを踏まえて、その質問の回答は動画配信サービスを選んだ回答を「1」、選んでない回答を「0」と記録した。他の質問に対して、回答の番号で記録した。

このように、相関分析を行うと、以下の結果となっている。

（３）分析結果

□　推薦の正確さと見る意欲に正の相関がある。

相関分析から、「推薦してくれた映画は、自分の感性にあっていると思いますか？」と「推薦してくれた映画は、見たいと思いますか？」の相関係数は 0.8 ほどあり、強い相関であることがわかった。

散布図を書いてみると、以下の通りである。

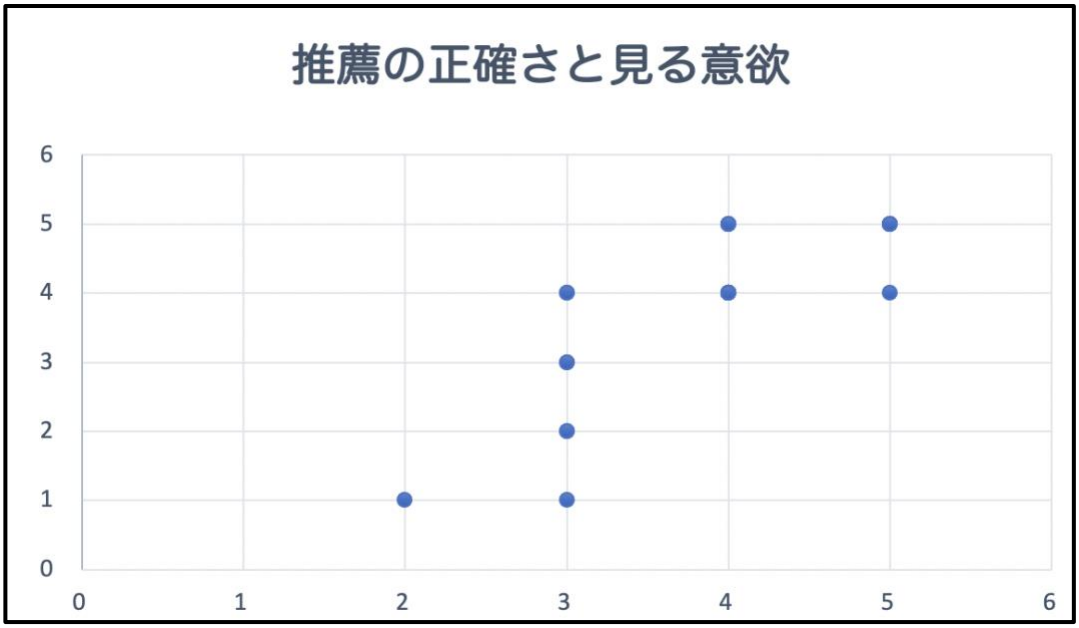


図 5-14 推薦の正確さと見る意欲散布図

Fig.5-4 scatter plot graph of recommendation accuracy and motivation

推薦の結果は正確なほど、その映画を見たくなることは非常に分かりやすいと思う。ユーザが見たい映画の感性を入力したので、正確に推薦できれば、ユーザの好みに合致する可能性が高い。

□　推薦の正確さと見る意欲はシステムを使用する意欲と相関がある。

このシステムを使う過程で、積極的な体験をしたので、システムを使いたくなることも理解しやすいと思う。

□　映画を見る頻度と映画を選ぶとき気持ちに対する重視度は正の相関がある。

アンケートの回答数が多いと言えないので、相関係数が普段より小さくても、ある程度の相関があると考えてもいいだろう。

この結果は予想通りで、自分のイメージだが、多く映画を見る人は、映画に対する拘りが多い。

□　映画を見る意欲と映画の見る頻度と相関している。
この結果は予想できなかったが、理解できる結果だと思う。
映画をよく見る人は、映画が好きな人の可能性が高くて、推薦してくれる映画に興味を持つ可能性が高い。

□　その他
年代とシステムを使用する意欲も少し相関であると見える。理由を推測してみると、映画を見ることが少ない、またシステムを使うとき、不便だと思うかもしれない。そのため、システムの対象層をよく映画を見る若者にした方がいいかもしれない。また、ユーザインタフェースをより使いやすく更新した方がいいかもしれない。
また、映画の感性判断は筆者（20 代）一人で判断した結果のため、年代別と使用意欲は負の相関である原因だと推測する。

5.7 考察

検証実験で、ユーザに実際に本映画推薦システムを体験してもらったこと、そしてアンケートから得た結果を分析したことからの感想、また、データ分析により、本研究の提案及び本システムの改善点をまとめる。
本システムはアンケートから、全体的に良い評価をもらった。システム全体の有効性を証明したと考える。そのため、本研究の必要性と本提案システムの有効性を検証することができたと考えられる。
そして、根本的機能「感性情報による映画の推薦」もポジティブな評価をもらったが、より正確に推薦できるように、推薦方法や機械学習の調整などを改善する必要もあると考える。
また、アンケート結果をデータ分析し、よく映画を見る人が映画を選択する感性をより重視し、若者が本システムに対する利用する意欲が高いなどの傾向がわかった。
この結果から、まず、本システムはよく映画を見る若者を対象にすれば、良い評価をもらいやすいと考える。また、よく映画を見る若者を対象にして、システムの UI デザインや映画感性データベースを拡大することで、映画の選択などの参考になれると思う。
また、映画をよく見る若者以外の人にシステムを利用させるために、年代別・映画の見る頻度別で、ユーザの特徴を抽出した上で、映画を推薦する方がいいと思われる。

第６章：結論と展望

6.1 結論

本研究は、映画を選ぶ際に、感情という嗜好に影響する重要な要素に着目し、このニーズを解決するための研究である。また、現在の映画推薦システム他の不足、例えば、新規映画が推薦しにくい、大量のユーザ情報がないと推薦できないなどを解決することを目指した。また、ユーザの映画鑑賞の視野を広げることを重視した。

本システムは、感性空間上での映画コンテンツの分類を畳み込みニューラルネットワーク（CNN）を用いた機械学習で行い、訓練済みの機械学習モデルを用い、自動的に映画の感性データを計算することにより、映画感性データベースを構築した。また、ピアソン相関係数で、ユーザの感性要求とデータベース中の映画をマッチングし、近い結果をユーザに提示する。

映画推薦システムの実現について、本研究は Flask を用い、ウェブアプリケーションの開発を行った。結果として、感情による推薦ができるシステムを実現することができた。コンテンツベースフィルタリングを用い、協調フィルタリングのコールドスタート、推薦の不透明、大量のユーザ情報がなければ推薦できないなど問題を避けられた。また、感情による推薦のため、ジャンルなどの映画物理属性に基づかずに、多様性のある推薦を行うこともできた。

また、検証調査を行い、システムの有効性が検証された。一番重要な「感性情報による映画の推薦する」機能の有効性は検証できた。また、ウェブアプリケーションの体験について、全体的に良い評価をもらった。

検証調査の結果から、データ分析も行った。年齢・映画の見る頻度の違いにより、本システムに対する体験が異なっているとわかった。これは、今後本システムを改善するための参考になると考える。

6.2 今後の展望及び課題

（１）映画の他の要素での機械学習検討

本研究では、感性に基づく映画の推薦システムの構築に向けて、映画の典型的な画面を使用した機械学習を行うことで、感性に基づく分類を行った。しかし、映画では映像だけではなく、音楽、台詞、物語展開のテンポなども、視聴者の感性に訴え感情を喚起する重要な要素である。今後の研究としては、映像に限らず、映画の音声、テンポなども踏まえて、映画を感性に基づいて、分類する方法について検討を行う予定である。これによって、より精度高く映画を分類し、利用者に対しより効果的な推薦を行うことができると期待される。

（２）感情の個人差を踏まえる推薦

映画による感情は大体共通しているが、個人差もある。

実験試験のアンケートのデータ分析から、ユーザの年齢、映画を見る頻度は本システムに対する体験に相関がある。

そのため、システムを使用前に、ユーザの年齢・映画の鑑賞する頻度で、先にユーザの分類を行う。

また、各分類のユーザの感情の尺度や映画の好みを踏まえる推薦を行い、より感情の個人差を踏まえる推薦の実現が期待される。

謝辞

まず、慶應義塾大学大学院システムデザイン・マネジメント研究科の小木哲朗教授には、心から感謝を申し上げます。指導教授として、2 年間多くの指導をいただき、本当にありがとうございます。

同研究科の春山真一郎教授には、副査としてご助言をいただき、深く感謝申し上げます。

また、小木研究室の武藤英樹先輩は、学会発表から、機械学習についての技術問題まで様々助けってくれました。立ち上げた仮想マシンのおかげで、COVID-19 に全く影響されずに、研究を進めました。深く感謝申し上げます。

そして、同研究室に所属しているメンバー、留学生及び同期の方々に感謝いたします。さらに、協力してくれた、検証実験及びアンケート調査に参加したくれた方々に感謝いたします。

最後、日本へ留学することを支えてくれる両親と祖父母に感謝を申し上げます。

2021 年 1 月

王　鳳超

参考文献

[1] 森田昌宏，速水治夫:情報フィルタリンクシステム，情報処理，Vol.37, No.8, pp.751–758 (1996).

[2] 林貴宏, 尾内理紀夫: Web 上のレビューを利用した映画推薦システム,人工知能学会論文誌 30 巻 1 号 SP1-E,(2015)

[3] Carlos A. Gomez-Uribe, Neil Hunt: The Netflix Recommender System: Algorithms, Business Value, and Innovation, (2015)

[4] 小野智弘, 黒川茂莉, 本村陽一, 麻生英樹: ユーザ嗜好の個人差と状況差を考慮した映画推薦システムの実現と評価,情報処理学会論文誌 Vol.49 No.1, (2008)

[5] Zhao, S.; Gao, Y.; Jiang, X.; Yao, H.; Chua, T.-S.; and Sun, X. : Exploring principles-of-art features for image emotion recognition. In Proceedings of the ACM International Conference on Multimedia, MM ’14, 47–56. New York, NY, USA: ACM, (2014).

[6] 金多賢, 北島宗雄, 李昇姫:映像に対する嗜好と感情反応・印象評価の関係, 日本感性工学学会論文誌 Vol.13 No.1(特集号)pp.181-189, (2014)

[7] Jana Machajdik, Allan Hanbury. : Affective image classification using features inspired by psychology and art theory. MM '10: Proceedings of the 18th ACM international conference on Multimedia, October 2010 Pages 83–92, (2010).

[8] 大出訓史，安藤彰男，今井篤，谷口高士:音楽聴取における“感動”の評価要因:感動の種類と音楽の感情価の関係， 情報処理学会論文誌，V ol.50，No.3，pp.1111-1121，2009.

[9] Philippot, P.: Inducing and assessing differentiated emotion- feeling states in the laboratory, Cognition and Emotion, 7(2), pp.171-193, 1993.

[10] James J. Gross, Robert W. Levenson: Emotion elicitation using films, Cognition and Emotion, 9(1), pp.87-108, 1995.

[11] 高橋靖(2000) 「セマンティックスコア法を用いた映画の構造表現」,デザイン学研究 Vol.47 No.4

[12] 池添剛・梶川嘉延・野村康雄(2001) 「音楽感性空間を用いた感性語による音楽データベース検索システム」, 情報処理学会論文誌Vol.42 No.12

[13] 梅崎利矢・千種健太郎・村木航介・苗村憲司(1998) 「レコメンダシステム及びその関連技術の比較検討と新システムの提案」

[14] 神嶌敏弘: 推薦システムのアルゴリズム(1), 人工知能学会誌 22 巻 6 号, (2007)

[15] Francesco Ricci, Lior Rokach, Bracha Shapira: Recommender Systems Handbook, Second Edition, (2015)

[16] 大杉直樹, 門田暁人, 森崎修司, 松本健一: 協調フィルタリングに基づくソフトウェア機能推薦システム, 情報処理学会論文誌, Vol.45, No.1,(2004)

[17] 高須賀清隆, 丸山一貴, 寺田実: 閲覧履歴を利用した協調フィルタリングによる Web ページ推薦とその評価, 情報処理学会研究報告, (2007)

[18] 木村篤信: 映像コンテンツ推薦の現状と今後, 映像情報メディア学会誌 Vol.68, No.4(2014)

[19] 服部しのぶ, 高木真一, 小館亮之, 冨永英義: 映像特徴に基づく自動映像分類システムの提案, オーディオビジュアル複合情報処理 36-4(2002)

[20] 吉井和佳, 後藤真孝, 駒谷和範, 尾形哲也, 奥乃博: ユーザの評価と音響的特徴との確率的統合に基づくハイブリッド型楽曲推薦システム, 情報処理学会研究報告 (2006)
[21] 富士谷康, 村尾和哉, 望月祐洋, 西尾信彦: コンテンツの多様性を考慮したクロスドメイン推薦, 情報処理学会論文誌, Vol57, No.10(2016)
[22] 土方嘉徳: 嗜好抽出と情報推薦技術，情報処理，Vol.48, No.9, pp.957–965 (2007).
[23] Michael J. Pazzani and Daniel Billsus, Content-Based Recommendation Systems, 2007
[24] Michael J. Pazzani, A Framework for Collaborative, Content-Based and Demographic Filtering, 1999
[25] Zeyu Cui, Feng Yu, and Shu Wu: Disentangled Item Representation for Recommender Systems, (2018)
[26] Libing Wu, Cong Quan, Chenliang Li, Qian Wang, Bolong Zheng: A Context-Aware User-Item Representation Learning for Item Recommendation, IEEE Transactions on knowledge and data engineering, (2017)
[27] 大西祐紀, 菅谷信介: コールドスタート問題解決のためのソフトクラスタリング活用の提案、情報処理学会代 79 回全国大会、(2017)
[28] Pariser, E.: The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think. Penguin Books (2012)
[29] Pariser, E.: The Filter Bubble: What the Internet Is Hiding from You. Penguin Books (2011)
[30] Maddi, S.R.: The pursuit of consistency and variety. In: R.P. Abelson, E. Aronson, W.J. McGuire, T.M. Newcomb, M.J. Rosenberg, P.H. Tannenbaum (eds.) Theories of Cognitive Consistency: A Sourcebook. Rand McNally (1968)
[31] 高田琢弘: 映像視聴によって喚起される感情の比較, Tsukuba Psychological Reasearch,(2013)

外部発表

（１）王鳳超、Tianniu Lyu、武藤英樹、小木哲朗:「機械学習を用いた感性情報に基づく映画のレコメンデーション」, ヒューマンインタフェース学会ヒューマンインタフェースサイバーコロキウム（2020 年 10 月）

付録

付録 1.機械学習を用いた感性情報に基づく映画推薦システムに関するアンケート

機械学習を用いた感性に基づく映画推薦システム体験してくれて、
ありがとうございます。
それでは、感想を聞かせてください！

・アンケートのデータは研究だけのために収集します。
・このアンケートはおよそ 1~3 分間かかります。

まず、あなたについて少し教えてください：

1. あなたの年代を教えてください。
   ○10 代以下　○20 代　○30 代　○40 代　○50 代　○60 代　○70 代以上

2. あなたの性別を教えてください。
   ○女性　○男性

3. あなたの映画を見る頻度を教えてください。
   ○あまり見ない　　○平均 1 ヶ月に 1 回～2 回
   ○平均 1 ヶ月に 3 回 5 回　　○平均 1 ヶ月に 6 回以上

4. あなたは映画を見る時よく使っているサービスを教えてください。（複数回答可）
   □映画館
   □動画配信サービス（Netflix、Amazon Prime Video、Hulu など）
   □家庭用 DVD・ブルーレイ
   □テレビ

これからは、感性に基づく映画推薦システムを体験した感想について：

5. 映画を選ぶとき、気持ちを考えた上で選ぶ必要があると思いますか？
   □そう思わない　□あまりそうおもわない　□どちらとも言えない　□ややそう思う　□そう思う

6. この推薦システムの推薦してくれた映画は自分の感性にあっていると思いますか？
   □そう思わない　□あまりそうおもわない　□どちらとも言えない　□ややそう思う　□そう思う

7. このシステムに推薦してくれた映画は見たいと思いますか？
   □そう思わない　□あまりそうおもわない　□どちらとも言えない　□ややそう思う　□そう思う

8. 今後この映画推薦システムを利用したいと思いますか？
   □そう思わない　□あまりそうおもわない　□どちらとも言えない　□ややそう思う　□そう思う

以上で終わります。ご協力ありがとうございます！:)

付録 2.　ソースコード

機械学習

```
import os
result_path = 'new_image/'
amusement_dir = os.path.join(result_path + 'amusement')
anger_dir = os.path.join(result_path + 'anger')
contentment_dir = os.path.join(result_path + 'contentment')
disgust_dir = os.path.join(result_path + 'disgust')
excitement_dir = os.path.join(result_path + 'excitement')
fear_dir = os.path.join(result_path + 'fear')
sadness_dir = os.path.join(result_path + 'sadness')

import tensorflow as tf
import keras_preprocessing
from keras_preprocessing import image
from keras_preprocessing.image import ImageDataGenerator

TRAINING_DIR = result_path
training_datagen = ImageDataGenerator(
      rescale = 1./255,
        rotation_range=40,
      width_shift_range=0.2,
      height_shift_range=0.2,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')

VALIDATION_DIR = result_path #"/tmp/rps-test-set/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    target_size=(150,150),
    class_mode='categorical',
   batch_size=126
)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    target_size=(150,150),
    class_mode='categorical',
```

```
    batch_size=126
)

model = tf.keras.models.Sequential([
    # Note the input shape is the desired size of the image 150x150 with 3 bytes color
    # This is the first convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu', input_shape=(150, 150, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    # The second convolution
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The third convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # The fourth convolution
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    # Flatten the results to feed into a DNN
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.5),
    # 512 neuron hidden layer
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(7, activation='softmax')
])


model.summary()

model.compile(loss = 'categorical_crossentropy', optimizer='rmsprop', metrics=['accuracy'])

history = model.fit(train_generator, epochs=500, steps_per_epoch=5, validation_data = validation_gener
ator, verbose = 1, validation_steps=3)
```

自動的に予告編から画像を抽出し、ファイルに保存する：

```
import os
file_path="flaskr/static/video/"
path_list=os.listdir(file_path)

path_name=[]

for i in path_list:
    if i.endswith(".mp4"):#mp4 文件
        path_name.append(i.split(".")[0])
    #若带有后缀名，利用循环遍历 path_list 列表，split 去掉后缀名

for file_name in path_name:
    # "a"表示以不覆盖的形式写入到文件中,当前文件夹如果没有"save.txt"会自动创建
    with open("flaskr/static/video/name.txt","a")as file:
        file.write(file_name+"\n")
        print(file_name)
    file.close()

def mkdir(path):
    folder = os.path.exists(path)

    if not folder:                #判断是否存在文件夹如果不存在则创建为文件夹
        os.makedirs(path)         #makedirs 创建文件时如果路径不存在会创建这个路径
        print ("---  new folder...")
        print("---  ok  ---")

    else:
        print ("---  folder exist!  ---")

for i in path_name:
    folder = file_path+i
    mkdir(folder)

import cv2
# 使用 opencv 按一定间隔截取视频帧，并保存为图片
def getPic(path):
    vc = cv2.VideoCapture(path) # 读取视频文件
    c = 0
    print("------------")
    if vc.isOpened(): # 判断是否正常打开
        print("yes")
        rval, frame = vc.read()
    else:
```

```
        rval = False
        print("false")

    timeF = 10000000  # 视频帧计数间隔频率

    while rval:  # 循环读取视频帧
        rval,frame = vc.read()
        #print(c,timeF,c%timeF)
        if (c % timeF == 0):# 每隔 timeF 帧进行存储操作
            #print("write...")
            cv2.imwrite(path.split('.')[0]+'/'+ str(c) + '.jpg', frame)  # 存储为图像
            print("success!")
        c = c + 100000
    cv2.waitKey(1)
    vc.release()
    print("================================")

for video in path_list:
    if video.endswith(".mp4"):
        video_path=file_path+video
        print(video_path)
        print(video_path.split('.')[0])
        getPic(video_path)

import openpyxl
from openpyxl import Workbook
from openpyxl.reader.excel import load_workbook

from openpyxl import Workbook

wb=Workbook()  #创建一个工作表
ws=wb.active   #ws 操作 sheet 页
ws1=wb.create_sheet('movieEmotion',0)
ws1.append(['movie_name',
            'amusement','anger','contentment',
            'disgust','excitement','fear','sadness'])

wb.save('flaskr/static/video/data/data100+.xlsx')

wb=load_workbook('flaskr/static/video/data/data100.xlsx')
ws1 = wb['movieEmotion']

from openpyxl import load_workbook

```

```
excel=load_workbook('flaskr/static/video/movieEmotionData.xlsx')

for i in path_name:
    pic_number = 0
    picNum=0
    #print(i)
    for m in os.listdir(file_path+i):
        if m.endswith(".jpg"):
            #print(i)
            pic_number += 1
            #pics.append(pic_number)
    #picNum=np.max(pics)
    #print(pic_number)

    print(i)
    print(pic_number)
    #picNumber(file_path+i)
    ws.append([i,pic_number])


wb.save('flaskr/static/video/movieEmotionData.xlsx')
```

自動的に画像の感性を判断する：

```
new_model = tf.keras.models.load_model('rps.h5')

new_model.summary()

from keras.preprocessing import image

img = image.load_img('flaskr/static/image/jurassic world fallen kingdom.jpeg', target_size=(150, 150))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)

images = np.vstack([x])

model = tf.keras.Sequential([new_model,
                             tf.keras.layers.Softmax()])
predictions = model.predict(x)
print(predictions)
np.argmax(predictions)

import os
file_path = 'flaskr/static/video/images_from_videos_100+/'
file_list = os.listdir(file_path)
print(file_list)
print(len(file_list))

file_list.remove('.ipynb_checkpoints')
print(file_list)
print(len(file_list))
#'.ipynb_checkpoints' 要删除不然下一步会卡住

import openpyxl
from openpyxl import Workbook
from openpyxl.reader.excel import load_workbook

from openpyxl import Workbook

wb=load_workbook('flaskr/static/video/data/data100+.xlsx')
ws1 = wb['movieEmotion']
for i in file_list:
    print(i, 'の結果は：')
    pics = [x for x in os.listdir(file_path+i+'/')
            if not x.startswith('.')] # 图片列表
    total_result=[]
    for pic in pics:
```

```
44.         img = image.load_img(file_path+i+'/'+pic, target_size=(150, 150))
45.         x = image.img_to_array(img)
46.         x = np.expand_dims(x, axis=0)
47.         images = np.vstack([x])
48.         predictions = new_model.predict(x)
49.         result=np.argmax(predictions)
50.         #print(result)
51.         total_result.append(result)
52.
53.     amusement=total_result.count(0)/len(total_result)*100
54.     anger=total_result.count(1)/len(total_result)*100
55.     contentment=total_result.count(2)/len(total_result)*100
56.     excitement=total_result.count(3)/len(total_result)*100
57.     disgust=total_result.count(4)/len(total_result)*100
58.     fear=total_result.count(5)/len(total_result)*100
59.     sadness=total_result.count(6)/len(total_result)*100
60.     ws1.append([i,amusement,anger,contentment,disgust,excitement,fear,sadness])
61.     print('recorded',i)
62.
63.     wb.save('flaskr/static/video/data/data100+.xlsx')
64. print('------------finished!---------------')
```

movie_recommender_system.py

```
# -*- coding =utf-8 -*-

app = Flask(__name__)
app.config['SECRET_KEY'] = 'hard to guess string'

bootstrap = Bootstrap(app)
moment = Moment(app)

class EmoForm(FlaskForm):
    amusement = StringField('input amusement(楽しさ） : ', validators=[DataRequired()]) # 1
    anger = StringField('input anger（怒り）:', validators=[DataRequired()]) # 2
    contentment = StringField('input contentment（安心）:', validators=[DataRequired()]) # 3
    disgust = StringField('input disgust（嫌悪）:', validators=[DataRequired()]) # 4
    excitement = StringField('input excitement（興奮）:', validators=[DataRequired()]) # 5
    fear = StringField('input fear（恐怖）:', validators=[DataRequired()]) # 6
    sadness = StringField('input sadness（悲しみ）:', validators=[DataRequired()]) # 7

    submit = SubmitField('Submit')


class FeedbackForm(FlaskForm):
    feedback = RadioField('Label', choices=[('Y', 'あっている！'), ('N', 'あっていない')])


@app.route('/', methods=['GET', 'POST'])
def instructions():
    return render_template('instructions.html')


@app.route('/instructions/cn', methods=['GET', 'POST'])
def instructions_cn():
    return render_template('instructions_cn.html')


@app.route('/recommender', methods=['GET', 'POST'])
def recommender():
    amusement = random.randint(20, 100)
    anger = random.randint(0, 20)
    contentment = random.randint(10, 100)
    disgust = random.randint(0, 10)
     excitement = random.randint(0, 100)
     fear = random.randint(0, 15)
```

```
43.     sadness = random.randint(0, 30)
44.
45.     form = EmoForm()
46.     if form.validate_on_submit():
47.         amusement = form.amusement.data  # 1
48.         form.amusement.data = ''
49.         anger = form.anger.data  # 2
50.         form.anger.data = ''
51.         contentment = form.contentment.data  # 3
52.         form.contentment.data = ''
53.         disgust = form.disgust.data  # 4
54.         form.disgust.data = ''
55.         excitement = form.excitement.data  # 5
56.         form.excitement.data = ''
57.         fear = form.fear.data  # 6
58.         form.fear.data = ''
59.         sadness = form.sadness.data  # 7
60.         form.sadness.data = ''
61.     emotion = [amusement, anger, contentment, disgust, excitement, fear, sadness]
62.     list_emotion = []
63.     for emotion in emotion:
64.         emotion = int(emotion)
65.         list_emotion.append(emotion)
66.         # 不能写 list_emotion=list_emotion.append(emotion),因为会返回空列表
67.
68.     wb = load_workbook('static/data100+.xlsx')
69.     ws1 = wb.get_sheet_by_name('movieEmotion')
70.     ws2 = wb.get_sheet_by_name('result')
71.     i = 0
72.     all_result_emo = []  # 保存所有电影与用户要求的查的列表
73.     second_emo_list = []  # 用于搜索第二个匹配结果的列表（去除最匹配结果
74.     third_emo_list = []  # 用于搜索第 3 个匹配结果的列表（去除最匹配结果
75.     forth_emo_list = []  # 用于搜索第 4 个匹配结果的列表（去除最匹配结果
76.     fifth_emo_list = []  # 用于搜索第 5 个匹配结果的列表（去除最匹配结果
77.     result_emo_data = []
78.
79.     def cosine(data1, data2):  # 余弦相似度 cosine コサイン類似度
80.         sum_data = data1 * data2.T  # ベクトルは data1 と data2
81.         denom = np.linalg.norm(data1) * np.linalg.norm(data2)
82.         # 归一化  结果を(0, 1)にする
83.         return 0.5 + 0.5 * (sum_data / denom)
84.
85.
86.     def pearson(data1, data2):  # pearson 皮尔逊相关系数
```

```
87.     avg1 = np.mean(data1)
88.     avg2 = np.mean(data2)
89.     sum_data = (data1 - avg1) * (data2 - avg2).T # ベクトルは data1 と data2
90.     denom = np.linalg.norm(data1 - avg1) * np.linalg.norm(data2 - avg2)
91.     # 归一化 結果を(0, 1)にする
92.     return 0.5 + 0.5 * (sum_data / denom)
93.
94. def differ(a, b): # 各项值差绝对值的和
95.     differ = a - b
96.     plus_differ = [abs(x) for x in differ] # 取绝对值
97.     return sum(plus_differ) # 每项的差相加的和
98.
99. for row_objects in ws1['B2':'H104']: # 搜索范围
100.    i = i + 1
101.    emo_value = []
102.    for cell_object in row_objects:
103.        emo_value.append(cell_object.value)
104.    data1 = np.mat(np.array(list_emotion)) # turn list into array
105.    data2 = np.mat(np.array(emo_value))
106.    a = np.array(list_emotion)
107.    b = np.array(emo_value)
108.    m = pearson(data1, data2)
109.    y = np.array(m)
110.    num_result = float(y)
111.    all_result_emo.append(num_result)
112.    second_emo_list.append(num_result)
113.    third_emo_list.append(num_result)
114.    forth_emo_list.append(num_result)
115.    fifth_emo_list.append(num_result)
116. result_movie_place = np.argmax(all_result_emo)
117. #    print('result dezhi', min(all_result_emo))
118. #    print('first:', result_movie_place, 'all list:', all_result_emo)
119. result_movie_name = ws1.cell(row=result_movie_place + 2, column=1).value # 搜索结果：电影名
120.
121. second_emo_list.remove(max(second_emo_list)) # 第二列表完成
122.
123. second_movie_num = max(second_emo_list)
124. for i in range(2):
125.
126.    third_emo_list.remove(max(third_emo_list))
127. third_movie_num = max(third_emo_list)
128. print('third has', len(third_emo_list))
129. for i in range(3):
130.    forth_emo_list.remove(max(forth_emo_list))
```

```
    forth_movie_num = max(forth_emo_list)
    for i in range(4):
        fifth_emo_list.remove(max(fifth_emo_list))
    fifth_movie_num = max(fifth_emo_list)

    str(all_result_emo)
    second_movie_place = all_result_emo.index(second_movie_num)
    third_movie_place = all_result_emo.index(third_movie_num)
    forth_movie_place = all_result_emo.index(forth_movie_num)
    fifth_movie_place = all_result_emo.index(fifth_movie_num)
    print('places:', second_movie_place, third_movie_place, forth_movie_place)
    second_movie_name = ws1.cell(row=second_movie_place + 2, column=1).value  # 搜索结果：第二个电影名
    third_movie_name = ws1.cell(row=third_movie_place + 2, column=1).value
    forth_movie_name = ws1.cell(row=forth_movie_place + 2, column=1).value
    fifth_movie_name = ws1.cell(row=fifth_movie_place + 2, column=1).value
    print('names:', result_movie_name, second_movie_name, third_movie_name, forth_movie_name)
    ip = request.remote_addr
    ws2.append([result_movie_name, second_movie_name, third_movie_name, forth_movie_name, fifth_movie_name,
                amusement, anger, contentment, disgust, excitement, fear, sadness,    ip])
    wb.save('static/data100+.xlsx')
    return render_template('index.html', form=form, emotion_require=emotion,    result_movie_name=result_movie_name)


@app.route('/show', methods=['POST', 'GET'])
def result():
    ip = request.remote_addr
    movie_emo_data = []
    wb = load_workbook('static/data100+.xlsx')  # open the excel
    ws1 = wb.get_sheet_by_name('movieEmotion')
    ws2 = wb.get_sheet_by_name('result')
    ws3 = wb.get_sheet_by_name('feedback')
    rows_for_this_ip = []

    for cell in list(ws2.columns)[12]:
        if cell.value == ip:
            #  print('dijihang:', cell.row)
            rows_for_this_ip.append(cell.row)
    row = max(rows_for_this_ip)
    #    print('ahahah', row)
    title = ws2.cell(row=row, column=1).value  # 读取最新记录在第一列的结果（搜索结果的电影名）
```

```
    title2 = ws2.cell(row=row, column=2).value  # 读取最新记录在第 2 列的结果（搜索结果第 2 匹配的电影名）
    title3 = ws2.cell(row=row, column=3).value  # 读取最新记录在第 3 列的结果（搜索结果第 2 匹配的电影名）
    title4 = ws2.cell(row=row, column=4).value  # 读取最新记录在第 4 列的结果（搜索结果第 2 匹配的电影名）
    title5 = ws2.cell(row=row, column=5).value  # 读取最新记录在第 4 列的结果（搜索结果第 2 匹配的电影名）

    image_name = title + '.jpeg'
    image2_name = title2 + '.jpeg'
    video1_name = title + '.mp4'
    video2_name = title2 + '.mp4'

    for cell in ws1['A']:
        if cell.value == title:
            # print(cell, cell.row, cell.column)
            result_row = cell.row  # 定位结果行
    print('result_row is :', result_row)

    for i in range(2, 9):
        movie_emo_data.append(ws1.cell(row=result_row, column=i).value)
    print('movie_emo_data is:', movie_emo_data)  # 搜索结果的各项情感值（画图用）

    form = FeedbackForm()
    feedback = None
    if form.validate_on_submit():
        feedback = form.feedback.data
    print('feedback is:', feedback)

    time = str(datetime.now())
    ws3.append([title, ip, time, feedback])
    wb.save('static/data100+.xlsx')
    return render_template('show_result.html', title=title, title2=title2,
                           title3=title3, title4=title4, title5=title5,
                           image_name=image_name, image2_name=image2_name,
                           video1_name=video1_name, video2_name=video2_name, form=form, feedback=feedback)

if __name__ == "__main__":
    app.debug = True
    app.run(host='0.0.0.0', port=5000)
```

base.html

```
{% extends "bootstrap/base.html" %}

{% block title %}movie recommender{% endblock %}
{% block head %}
{{ super() }}
<link rel="shortcut icon" href="{{ url_for('static', filename='yeah.jpeg') }}" type="image/x-icon">
<link rel="icon" href="{{ url_for('static', filename='yeah.jpeg') }}" type="image/x-icon">
{% endblock %}

{% block navbar %}
<div class="navbar navbar-inverse" role="navigation">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
                <span class="sr-only">Toggle navigation</span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            <a class="navbar-brand" href="/recommender">Movie Recommender</a>
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li><a href="/">Instructions</a></li>
                <li><a href="/questionnaire" >Questionnaire</a></li>
            </ul>
        </div>
    </div>
</div>
{% endblock %}

{% block content %}
<div class="container">
    {% block page_content %}{% endblock %}
</div>
{% endblock %}

{% block scripts %}
{{ super() }}
{{ moment.include_moment() }}
{% endblock %}
```

index.html

```
{% extends "base.html" %}
{% import "bootstrap/wtf.html" as wtf %}
{% block title %}movie recommender{% endblock %}
{% block page_content %}
<style>
    h5{color:gray;}
    .a {text-transform:capitalize;} /*首字母大写 */
</style>
<h4>Please input your emotion here(1~100):</h4>
<h5>感性情報を入力してください(1~100):</h5>
{{ wtf.quick_form(form) }}
<div class="page-header">
    <!--
    <h1>Hi, {% if name %} {{ name }} {% else %} stranger{% endif %}</h1>
    !-->
    <h3>Maybe you want to watch:{% if result_movie_name %}<br>
        <a href="/show" title="click to see more"/><i class="a"><center>{{ result_movie_name }}</i></
a>
        {% else %}nothing{% endif %}</h3>
</div>


{% endblock %}
```

show_result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
{% extends "base.html" %}
{% block title %} Recommender Result {% endblock %}

<style>
    .a {text-transform:capitalize;} /*首字母大写 */
</style>

</head>
{% block page_content %}

<body>
<div><center>
   <h3 class="a"><a href="https://www.google.com/search?q= {{ title }}"> {% if title %}
        {{ title }}
       {% else %}nothing{% endif %}</a></h3>
   <img src="static/image/{{ image_name }} " width="270" height="400" alt="{{title}}"/>
<br>
   <video width="500" height="280" controls>
      <source src="static/video/{{ video1_name }}" type="video/mp4">
您的浏览器不支持 video 标签。
</video>
</div>

<div id="feedback">
<h4>この結果はあなたの感性にあっていますか？</h4>
<h5>Is the result match the emotion?<h5>
<form method="post">
   {{ form.hidden_tag() }}
   {{ form.feedback }}
   <input type="submit">
</form>
</div>

<div id="second result">
<h4 class="a">maybe you also want to watch:</h4>

   <a href="https://www.google.com/search?q= {{ title2 }} movie">
      <img src="static/image/{{ image2_name }} " width="135" height="200" title="{{title2}}" alt="{{title2}}"/>
```

```
    </a>
    <a href="https://www.google.com/search?q= {{ title3 }} movie">
    <img src="static/image/{{ title3 }}.jpeg" width="135" height="200" title="{{title3}}" alt="{{title3}}
"/>
    </a>
    <a href="https://www.google.com/search?q= {{ title4 }} movie">
    <img src="static/image/{{ title4 }}.jpeg " width="135" height="200" title="{{title4}}" alt="{{title4}
}"/>
    </a>
    <a href="https://www.google.com/search?q= {{ title5 }} movie">
    <img src="static/image/{{ title5 }}.jpeg " width="135" height="200" title="{{title5}}" alt="{{title5}
}"/>
    </a>
</div>

{% endblock %}
</body>
</html>
```