Bachelor of Science in Computer Science & Engineering

A Framework for Designing a Movie Recommendation System Based on User Preferences, Movie Ratings and Comments

by

MD. Ifte Kharul Alam

ID: 1504029

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

May, 2021

A Framework for Designing a Movie Recommendation System Based on User Preferences, Movie Ratings and Comments

Submitted in partial fulfilment of the requirements for
Degree of Bachelor of Science
in Computer Science & Engineering

by

MD. Ifte Kharul Alam

ID: 1504029

Supervised by

Md. Shafiul Alam Forhad

Assistant Professor

Department of Computer Science & Engineering

Chittagong University of Engineering & Technology (CUET)

Chattogram-4349, Bangladesh.

May, 2021

The thesis titled '**A Framework for Designing a Movie Recommendation System Based on User Preferences, Movie Ratings and Comments'** submitted by ID: 1504029, Session 2019-2020 has been accepted as satisfactory in fulfilment of the requirement for the degree of Bachelor of Science in Computer Science & Engineering to be awarded by the Chittagong University of Engineering & Technology (CUET).

Board of Examiners

Declaration of Originality

This is to certify that I am the sole author of this thesis and that neither any part of this thesis nor the whole of the thesis has been submitted for a degree to any other institution.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. I am also aware that if any infringement of anyone's copyright is found, whether intentional or otherwise, We may be subject to legal and disciplinary action determined by Dept. of CSE, CUET.

I hereby assign every rights in the copyright of this thesis work to Dept. of CSE, CUET, who shall be the owner of the copyright of this work and any reproduction or use in any form or by any means whatsoever is prohibited without the consent of Dept. of CSE, CUET.

MD. Iftekharul Alam

Signature of the candidate

Date: 24.05.2021

Acknowledgements

The success and final outcome of this thesis required a great deal of guidance and assistance from a large number of people, and I consider myself extremely fortunate to have received this guidance and assistance throughout the course of completing my thesis. It's been an enlightening experience on both a professional and personal level. All that I have accomplished is solely as a result of such oversight and assistance.

Above all, I want to express my gratitude to Allah for enabling me to successfully complete this thesis. Following that, I would like to express my heartfelt appreciation to Md. Shafiul Alam, Assistant Professor, Department of Computer Science and Engineering, Chittagong University of Engineering and Technology (CUET), for his guidance, encouragement, and unwavering support throughout my thesis work. I am indebted to him for his numerous critical questions, which forced me to consider things from new angles, and for his unwavering support throughout the process.

I am indebted to all members of the department's teaching staff for their constant encouragement, support, and guidance.

Finally, I want to express my gratitude to my father and mother for their unwavering love, support, encouragement, and contribution to every aspect of my life and academic career over the years.

Abstract

Recommender System is a tool that allows users to find and overcome the overload of information. It forecasts users' interests and makes recommendations based on their interest model. As the value of customized services grows, numerous studies on personalized recommendation systems are being conducted. Typically, data filtering and recommendation algorithms employ a variety of approaches and algorithms. The majority of state-of-the-art recommender systems take a content-centric approach but frequently miss the nature of users' desires. As a result, the primary objective of this work is to combine recommendation and sentiment analysis in order to provide users with the most accurate recommendations possible. This study's collaborative filtering, which reflected user review sentiment scores, was verified to be more accurate than collaborative filtering using the traditional method, which reflects only user rating data.

The results analysis reveals some intriguing findings about the impact of incorporating sentiment analysis into a collaborative filtering-based recommendation technique.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The introduction of Recommender Systems has changed the way people search for information, products, and even other people. Recommender Systems analyze patterns of behavior to predict what an individual might prefer from a collection of items he or she has never encountered. Over the last 15 years, the technology underlying recommender systems have evolved into a robust collection of tools that enables researchers, users, and practitioners to create effective recommender systems. Research in the field of recommendation systems has continued for several decades, but due to the abundance of practical applications and the rich area of problem the interest remains high. Many of these systems are being implemented and used, such as the Amazon.com book recommendation system, MovieLens.org movies, CDs at CDNow.com from Amazon.com etc.

Each user has unique preferences and interests. Additionally, even the taste of a single user can vary significantly depending on various factors, such as the user's mood, the season, or the type of activity. For instance, the type of music one prefers to listen to while exercising is quite different from the type of music one prefers to listen to while cooking dinner. A second issue that recommendation systems must address is the exploration vs. exploitation dichotomy. They must venture into new domains in order to learn more about the user while maximizing what is already known about the user. Since it is simply not possible to view each piece of information, it is difficult to find and choose information which meets each user's preferences. Numerous algorithms could be applied to data in order to predict a user preference. Therefore, the novel aspect of this work is that it combines sentiment analysis and recommendation using collaborative filtering to create a unique and functional recommender system.

1.2 Framework/Design Overview

The proposed framework consists of two components. The first component is responsible for analyzing user reviews and inferring ratings from them, while the second one is a collaborative filter that generates item recommendations based on the ratings inferred.

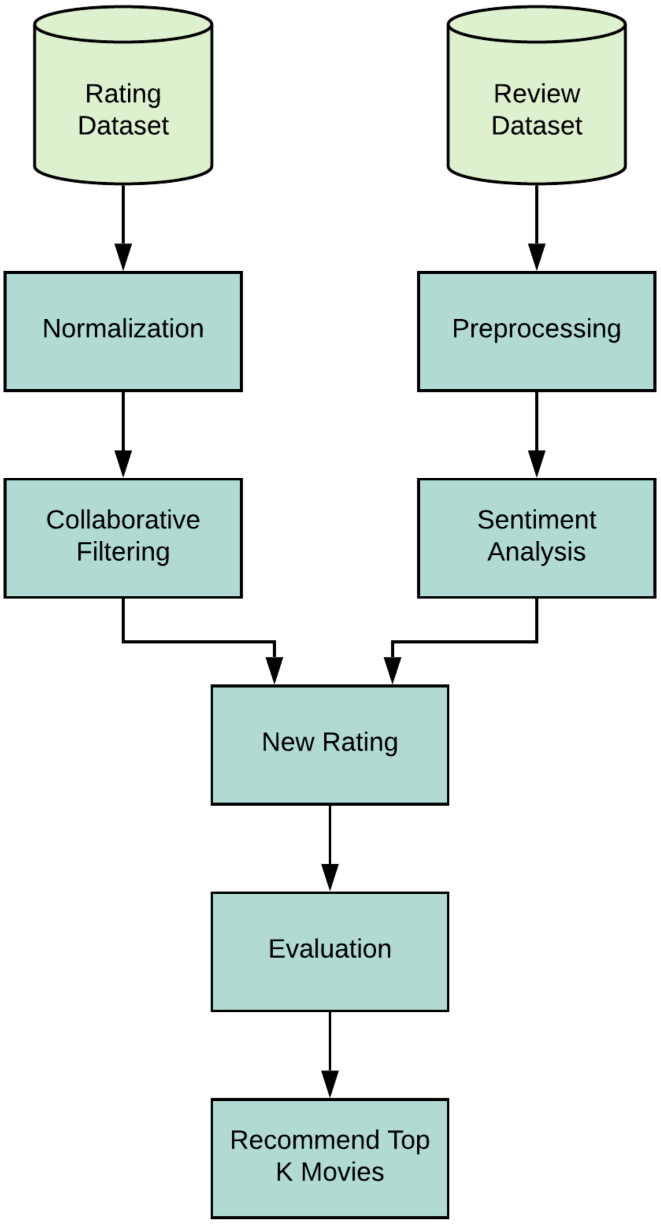


Figure 1.1: Overview of the proposed framework

In Figure 1.1 depicts an overview of the proposed framework. The following subsections provide further details about the rating inference component. It includes four major steps namely data preparation, review analysis, opinion dictionary construction and rating inference. Our discussions only focus on the rating inference component because, as noted, ratings inferred from user reviews can be fed into existing CF algorithms.

1.3 Difficulties

Recommendation is a critical component of our online lives. One can get lost in a network of data if they don't have any recommendations. Movies are a significant source of entertainment as well. The majority of the films we watch are those that have been suggested to us by others. Everybody has a favorite genre of film. As a result, a movie recommendation system will boost a rental and sale shop's sales. There are various challenges faced by Recommendation System. These challenges are Cold Start problem, Data Sparsity, Scalability.

1. Cold Start Problem: It requires a certain number of users in the framework to find a match. For example, if we try to locate a similar user or object, we compare it to the collection of existing users or entities. At first, a new user's profile is empty because he has not rated any items. The system has little idea about his preferences, making it impossible for the system to provide him with recommendations on any item. The same may be said about a new object since it has not been rated by any consumer because it is new to the user. Any of these issues may be addressed with the use of hybrid approaches.[1]

2. Data Sparsity: The user-generated or ranking matrix is extremely sparse. It's very difficult to locate users who have scored identical objects since the majority of users do not rate items. As a result, locating a group of consumers who rate the objects becomes more difficult. It's very difficult to make recommendations because you have no knowledge regarding a customer [2].

3. Scalability: Collaborative Filtering makes use of a large volume of data in order to improve reliability and requires a greater range of resources. When data

becomes increasingly larger, analysis becomes prohibitively costly and unreliable as a consequence of this Big data problem [3].

1.4 Applications

The term "recommendation system" covers a wide range of topics and is utilized in various fields. Since people use tips to save time, they are essential in a variety of fields. It has a broad range of real-world applications, including entertainment, e-commerce, services, and social media [4]. In the entertainment industry, feedback systems are commonly utilized while viewing films, listening to songs, or watching any television show. When it comes to e-commerce, Amazon is the world's biggest retailer. Some use it to purchase books, others to purchase kitchen goods or other products, and even others to purchase clothes. Thus, the whole planet is relying on these E-commerce websites for one or more of their functions. Other e-commerce websites, such as Flip cart, eBay, Myntra, and Shop clues, also provide suggestions. Additional implementations in recommendation systems include the following:

- **Film Recommendation:** Netflix utilizes an algorithm to suggest films based on a user's interests. Other recommendation-based sites include Hotstar, sony liv, voot, and ALTBalaji [5].

- **Music Recommendation:** Pandora creates a radio station based on your preferences. It recommends other songs based on the properties of the songs. Spotify, JioSavan, and Gaana are additional mediums in this area that indicate music recommendations.

- **News:** Google News, Apple News (integrated with iOS and macOS), Flipboard, Feedly, Tweet Deck, Pocket, Mix, Zig, and News360 are only a few of the apps that offer news recommendations. These all recommend news, blogs, blog posts, and material from top publishers, among other things [6].

- **Fashion:** People may purchase a variety of apparel pieces. Various shopping places such as Myntra, Amazon, Club Factory, SHEIN, Lime Road, Flip Cart, and others are included in this segment [7].

- **Transport service:** Recommendation aids in this area by recommending different travel places to ensure a smooth trip. This involves Road trippers, which makes it simple to schedule every road trip. Users will enter their travel arrangements into Hooper, and the software can inform them when the best time to book their flight is [8].

1.5 Motivation

A possible challenge of the knowledge overload has been generated by the exponential growth in digital information and the number of visitors to the internet which prevents timely internet access to things of interest. Information retrieval frameworks including Google, Devil Finder, and Altavista also partly solved the issue, but prioritization and personalization of information (where a device maps accessible material to a user's needs and preferences) were missing. This has resulted in a higher-than-ever market for recommender systems.

Recommender systems are knowledge filtering systems that address the issue of information overload [9] by extracting critical information fragments from a vast volume of dynamically produced data based on the user's needs, interests, or observable behavior [10]. Thus recommendation systems assist consumers in finding and selecting products (e.g., books, movies, restaurants) from across the vast array of options accessible on the internet or in other electronic content outlets. They provide the consumer with a limited collection of items that are well tailored to the description, given a broad set of items and a description of the user's needs. Based on the user's profile, a recommender framework will determine whether a specific consumer will like an object or not. Both market providers and consumers profit from recommender schemes [11]. They lower the costs of searching for and sorting movies in an online platform. Recommendation mechanisms have also been shown to increase the efficiency and method of decision-making [12]. Recommender schemes increase sales in an e-commerce environment because they are an efficient way of selling more goods [11]. Recommender schemes in research repositories assist consumers by helping them to go through index queries.

As a result, the importance of using effective and reliable recommendation techniques within a framework that provides consumers with appropriate and dependable recommendations cannot be overstated. Furthermore, it takes into account a variety of variables in order to build customized lists of valuable and interesting material for each user/individual. Recommendation systems are Artificial Intelligence-based algorithms that scan all available choices and generate a personalized list of things that are important and applicable to a specific person. These recommendations are focused on their profile, search/browsing background, what other people with similar characteristics/demographics are viewing, and the likelihood of watching certain movies. This is accomplished using data-driven statistical models and heuristics.

1.6 Contribution of the thesis

Thesis or research work is carried out to accomplish a specific set of objectives, such as defining a new methodology or improving an existing one. The primary objective of this thesis was to enhance the movie recommendation system. This thesis makes the following primary contribution:

1. We propose a recommendation system by combining collaborative filtering and sentiment analysis.

2. Sentiment analysis is used to boost up this recommendation system.

3. A detailed analysis of proposed recommendation system is presented through extensive experiment. Finally, a qualitative as well as quantitative comparison with other baselines models is also demonstrated.

1.7 Thesis Organization

The rest of this thesis report is organized as follows:

- Chapter 2 gives a brief summary of previous research works in the field of Movie recommendation system.

- Chapter 3 describes the proposed methodology for the Movie recommendation system. The proposed framework details the sentiment analysis process, its implementation, and a collaborative filtering technique. Implementation of a collaborative filtering process, draft recommended list
- Chapter 4 provides the description of the working dataset and analysis of the performance measure for the proposed framework.
- Chapter 5 contains the overall summary of this thesis work and provides some future recommendations as well.

1.8 Conclusion

In this chapter, an overview of movie recommendation system is provided. Along with the difficulties, the summary of the movie recommendation framework is described in this chapter. The motivation behind this work and contributions are also stated here. In the next chapter, background and present state of the problem will be provided.

Chapter 2

Literature Review

2.1 Introduction

In our lifetime, the recommending systems took place more and moreover the last decades with the growth of Youtube, Amazon, Netflix, and many other services like these. The recommending systems today are inevitable in our everyday online journeys from e-commerce (suggests buyers articles that might interest them) to online advertisements (suggests the right user contents, matching their preferences).

In the broadest sense, recommender systems are algorithms that aim to provide users with relevant items (items being movies to watch, text to read, products to buy, or anything else depending on industries).

Data on movies, in particular, are a common type of data that can be used to verify human emotion indicators and personalized urban services. As a result, research into recommendation systems based on extensive data analysis of movies has been steadily increasing. The public's preferences for products and movies are studied by categorizing emotions expressed in product reviews, product use reviews, and movie reviews written by anonymous people as negative or positive. It is used in various recommendation system areas appropriate for the user propensity by combining the analyzed preferences with personalized information.

Both service providers and users benefit from recommender systems [11]. Additionally, recommendation systems have been shown to enhance the decision-making process and quality [12]. In scientific libraries, recommender systems assist users by enabling them to conduct searches beyond the catalog. Thus, the importance of employing efficient and accurate recommendation techniques within a system capable of providing users with relevant and trustworthy recommendations cannot be overstated.

2.2 Related Literature Review

Numerous studies have already been conducted on the movie recommendation system. A recommender system is defined as a strategy for users to make decisions in complex information environments [13]. Additionally, recommender systems were determined from an E-commerce perspective as a tool that assists users in searching through records of knowledge that are relevant to their interests and preferences [14]. The term "recommender system" refers to a technique for assisting and augmenting the social process of making choices based on the recommendations of others when one lacks sufficient personal knowledge or experience with the alternatives [15]. By providing users with personalized, exclusive content and service recommendations, recommender systems alleviate information overload. Numerous approaches to developing recommendation systems have been developed recently [16], [17], [18]. These approaches can utilize collaborative filtering, content-based filtering, or hybrid filtering. Collaborative filtering is the most developed and widely used type of filtering technique. Collaborative filtering recommends items by identifying other users who share a similar taste; it then utilizes their judgment to make recommendations to the active user. In a variety of application areas, collaborative recommender systems have been implemented. GroupLens is a news-based architecture that aids users in locating articles in a massive news database [19]. Ringo is a social information filtering system for online users that uses collaborative filtering to create user profiles based on their album ratings. Amazon's recommendation engine makes use of topic diversification algorithms [20]. The system overcomes scalability issues by offline generating a table of similar items using an item-to-item matrix. The system then recommends similar products online based on the user's purchase history.

On the other hand, content-based techniques map user characteristics to content resources. Unlike collaborative techniques, content-based filtering techniques typically make predictions based on the user's information and ignore contributions from other users [21], [22]. Fab heavily relies on user ratings to generate a training set and is an example of a content-based recommender system. Letizia [23] is another system that utilizes content-based filtering to assist users in locating information on the Internet. The system makes use of a user interface to assist users in their Internet browsing; it is capable of tracking a user's browsing pattern to predict which pages they may be interested in. Pazzani et al. [24] created an intelligent agent that uses a naive Bayesian classifier to attempt to predict which web pages will be of interest to a user. The agent enables users to provide training instances by categorizing various web pages as hot or cold. Jennings and Higuchi [25] describe how a neural network can be used to model a user's interests in a Usenet news environment.

Despite their success, these two filtering techniques have several limitations. Several issues with content-based filtering techniques include insufficient content analysis, overspecialization, and data sparsity. Additionally, collaborative approaches have issues with a cold start, sparsity, and scalability. These issues frequently impair the quality of recommendations. To address a few of the issues identified, hybrid filtering has been proposed [26], [27]. Hybrid filtering combines two or more filtering techniques in novel ways to improve the accuracy and performance of recommender systems. These techniques combine two or more filtering approaches in order to maximize their advantages while minimizing their disadvantages [28]. They are classified into weighted hybrids, mixed hybrids, switching hybrids, feature-combination hybrids, cascade hybrids, feature-augmented hybrids, and meta-level hybrids based on their operations [29]. Today, collaborative filtering and content-based filtering approaches are widely used, with the results of their prediction being combined or the characteristics of collaborative filtering being added to content-based filtering, and vice versa. Finally, a general unified model could be developed that incorporates both content-based and collaborative filtering properties. The data sparsity and cold-start issues were addressed in [30] by combining item ratings, features, and demographic information in a cascade hybrid recommendation technique. Ziegler et al. [31] proposed a hybrid collaborative filtering approach based on the generation of profiles via inference of super-topic score and topic diversification to address the data sparsity problem associated with CF recommendations. Ghazantar and Pragel-Benett [30] also propose a hybrid recommendation technique that utilizes the content-based profile of an individual user to identify similar users for the purpose of making predictions. Sarwar et al. [32] used a combination of collaborative filtering and an information filtering agent. The authors propose a framework for integrating content-based and collaborative filtering agents in this article. Numerous applications employ a hybrid recommender algorithm as a result of the new user problem posed by content-based filtering techniques and the average user problem posed by collaborative filtering [33]. Cunningham et al. [34] proposed a straightforward method for combining content-based and collaborative filtering. Konstas et al. [35] proposed a music recommendation system that combined tagging information, play counts, and social relationships. Lee and Brusilovsky [36] used social information embedded in a collaborative filtering algorithm to determine the number of neighbors that can be automatically connected on a social platform. Condiff et al. [37] proposed a Bayesian mixed-effects model that incorporates user ratings, user and item characteristics into a unified framework. The related works and Various recommendation systems are surveyed in following section:

2.2.1 Content-Based Filtering

Content-Based Filtering is also known as CBF. Based on the user's previous experiences, this filtering recommendation. For example, If a user enjoys action films, the system presents him with related action films with better scores. If the user only likes political content, the system will provide him or her with political ones. We do not face the new challenges posed by content-based Filtering. It is non-interactive. It's about one specific user. User-based Filtering first checks his preferences and then presents the content. It deals only with single ideas and predictions for one person. So if we're talking about movies, the user-centered technique looks at the user's rating. The algorithm analyzes the user ratings for movies within the Genre filter. Using the analysis suggests movies for the user based on his/her taste It demonstrates how Content-Based Filtering works.

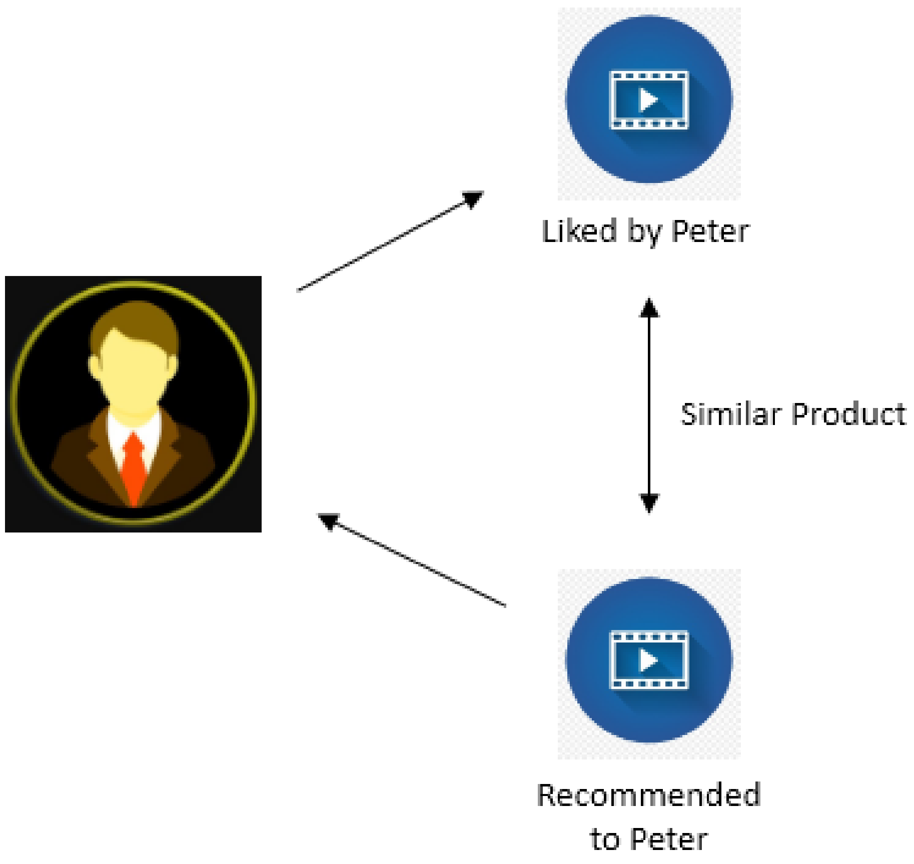


Figure 2.1: Content Based Filtering

In figure 2.1, Geometric Shapes shows the process of content filtering. The content-based filtering whole process is shown by giving an example of Geometric Shapes [24]. These shades are liked by the user. Today user profiling is performed. This data was obtained from the item profile. As we can see in the item analysis, the user likes the Circle and Triangle of color. Now we can match this user's shape with the list of conditions. We have a blue pentagon and two yellow circles in the shapes collection. Thus, here, the system selects which form (s) matches the user profile. Therefore, blue color for the user's interests. Jieun Son and Seoung Bum Kim proposed Content-based filtering for recommendation systems using multiattribute networks that contain attribute information about items [25]. Most attributes are based on network analysis, and all-specific problems are avoided. Results show that issues like sparsity and scalability are also solved when they are applied with the following: Linked Content the 100 human guinea pigs are used to conduct the experiments to find better means of finding the information In the paper, it was Providing Entertainment by Content-based Filtering and Semantic

Reasoning in Intelligent. Recommender Systems by Yolanda Blanco-Fernandez et al. solved the overspecialization problem [26]. If there is a hidden semantic association between the customer and the data, Spreading Activation is applied. It is irrelevant to supply user-specific recommendations. The content-oriented strategy works well. It helps users discover unique tastes. There are no terrible raters. Also, it provides explanations for the items we recommend. Identifying any particular feature like a picture or film may be a challenge. A general over-specialization problem outside recommendations are not listed There isn't enough information about that item to properly document it.

2.2.2 Collaborative Filtering

Tapestry presents a new option but with many issues to overcome. There have been systems such as Grouperformer and Grouper that do this for some time now. At present, many e-based businesses utilize Amazon, CDnow, Drugstore, and Moviefinder. The world is full of data. Today, everyone is pressed for time and does not have the luxury of picking through thousands of items that they like. Filtering user preferences is one of the methods of providing relevant information. Collaborative Filtering is a well-known method for item-based recommending. This method works by cross-referencing users' choices. It begins by identifying the neighbor and then predicts what will happen. This can be many users. It looks for similar users in the users' list. However, the connection between users and products is revealed based on product ratings. This way then results occur. This strategy takes ratings given by the user for any item from the large catalog of item catalog of ratings given by the user. This large catalog is referred to as a user-item matrix [28].

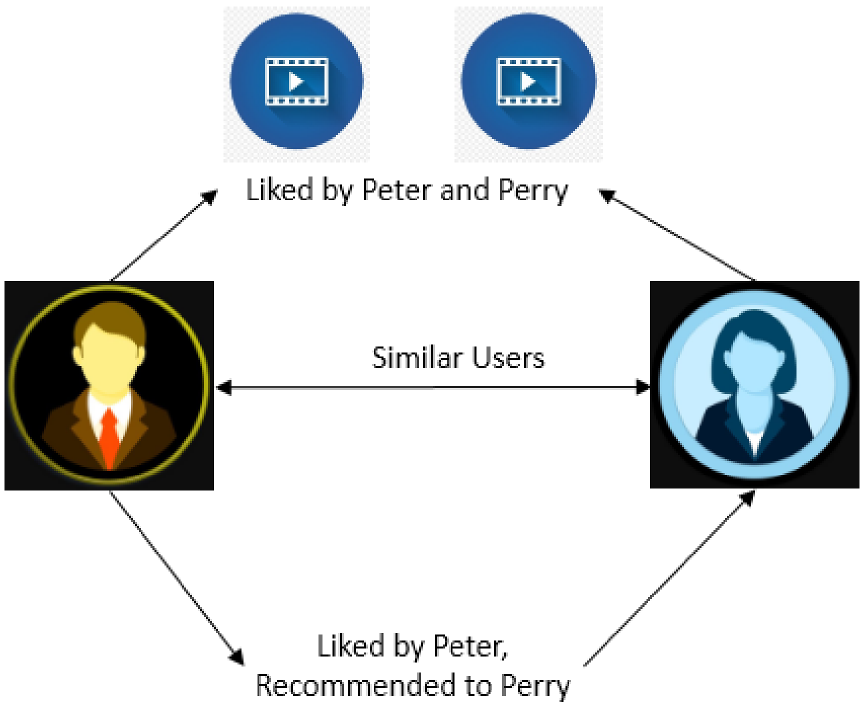


Figure 2.2: Collaborative Based Filtering

The figure 2.2 demonstrates collaborative filtering with an example. According to the visual depiction, users 1, 2, and 3 have rated movies based on their interests. As a result, a similarity model is applied, it is determined if it is possible to recommend movies for user 3.

Collaborative Filtering is further divided into two types. They are memory based and model based methods [32], [33].

2.2.2.1 Memory Based Method

This method is also known as the Neighborhood-based approach [34]. The memory-based method uses similarity measures calculated from explicit user rating to find a neighbor and generate predictions [35], [36]. This approach is enjoyable for everyone. To analyze the user's interests, we search for items that he/she is interested in. Utility matrix is searched for in this way. For purposes of getting similar user predictions, this is a systems-based approach, So in this case, the unknown rating can be calculated using the user rating matrix (utility matrix). At last recommendation can be given [38]. Memory based approach is further classified into two types. User based approach and Item based approach [39].

2.2.2.2 Model Based Method

In the Model-based method, the ratings of each user are used to predict the expected values of nonrated items. Mainly, a machine learning or data mining algorithm is used here. The model is built on the ratings provided by the users for any item. To get the data from the matrix Now the model has been created using the given data to predict for the users; it is additionally categorized into different categories in the number Association mining, Artificial Neural Network, Perceptronian, decision tree model-based research is being done

2.2.3 Hybrid Filtering

This filtering is an information filtering system that takes ratings of the movies as input from the users and then applies the collaborative and content-based filtering and generates a recommendation list [XMR or collaborative filtering, can work at that point Content and collaborative filtering go hand in hand when hybrid filtration is used a significant obstacle in collaborative filtering To start, we apply content-based filtering and then apply collaborative filtering This will solve the issue.

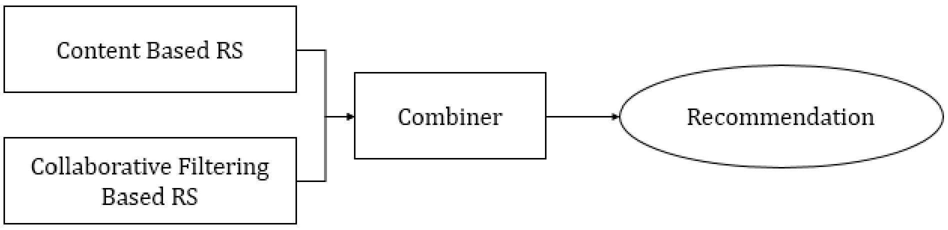


Figure 2.3: Hybrid Filtering

There are various categories of hybrid filtering. Given below:

2.2.3.1 Weighted hybridization

In this technique of hybridization, at first, outcomes of the items which are recommended are generated from the list of the recommendation techniques used in the particular system. The system named P- Tango is known as Personalized Tango [40], used this hybrid concept. This consists of front, backend, and database. The user accesses the front and downloads a web app to make predictions. Generally, it is used To summarize, allow for collaborative and content filtering, but apply individual weights for the results.

2.2.3.2 Switching hybridization

It has already been done. It is done based on the current state of the system. So criteria have been implemented to switch between the two systems. A ramp-up technique is typically is used to avoid. But both collaborative and content-based filtering face new user problems. This technique was introduced by a method called Daily Learner, which processes content-based tagging at startup. It is employed to handle cold-start issues in this manner. They used the "content-based" and "demographic" approach to contend with the "cold start".This switching hybridization method is used by [41] to deal with different cold start problems. The author used content-based CAMF-CC and demographic-based CAMF-CC to provide a way to cope up with a cold start problem.

2.2.3.3 Cascade hybridization

In this technique, the recommendation by one method is generated, and that recommendation results are refined or filtered by another recommendation technique to improve the recommendation system. A music recommender system developed by [42]. Using this technique, one recommendation is generated, and the results are refined or filtered to improve the system's recommendation. [42] is built for musicians. It acts as a middleware system for audio and music production studios. Content-based is used and crowd-sourced to provide recommendations. The system allows the music's genre to be specified by the user as well as considering previous user preferences. As well as a cascaded knowledge-based and collaborative recommender.

2.2.3.4 Mixed hybridization

As the name suggests, it employs multiple recommenders to provide numerous recommendations. When recommendations are desired, this is the method of getting them—an agent-based recommender system to gather basic site information on site usage and follow with collaborative filtering. Until the user is found, this system provides relevant pages via content-based and peer-suggested filtering.

2.2.3.5 Feature Combination

This feature starts with one technique and then emulates another. Basically, this consists of a merger and content-based filtering. This way, features from different recommender engines are combined and fed into one single recommendation algorithm [43].

2.2.3.6 Feature Augmentation

The outputs of one recommender system are used as inputs to another. The first recommender system takes in additional information and produces better recommendations. Now, the second method incorporates that information and produces further recommendations. Using some of the initial recommender features yields better results.

2.2.3.7 Meta Level

This is another way to combine two programs to produce recommendations. Internally or entirely generated by the recommender engine, it is applied to the other components of the system. It can be a common misconception that they are synonyms for one another. in feature augmentation, available data are added to the model, and then some to that generated by the system. Providing more data is unnecessary. It handles cold starts, sparsity, and gray color problems. Complexity also plays a big role in implementation.

2.2.4 Similarity Measures

There are various similarity measures to find out similarity between user and item. Some of them are discussed here.

2.2.4.1 Jaccard Similarity

It is the ratio of common items rated by the user to the total number of items rated by both the users. The formula to calculate Jaccard Similarity is given below [44].

$$sim(u,v)^{Jaccard} = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} \tag{2.1}$$

Here I_u and I_v are the set of item rated by the user u and v respectively.

2.2.4.2 Cosine Similarity

Cosine similarity [45], [46] is the measure of similarity between two non-zero vectors in the inner product space. It measures the angle between these two vectors. A cosine of two non-zero vectors can be calculated using dot products of these two vectors:

$$u.v = ||u||.||v||.cos\theta \tag{2.2}$$

Cosine similarity is particularly used in positive space where the result is efficiently bounded in [0, 1]. Thus for two given vectors u and v, the cosine similarity, $cos\theta$ can be computed as the combination of dot product and magnitude of the vectors:

$$sim(u,v)^{cosine} = cos\theta = \frac{\sum uv}{||u||.||v||} \tag{2.3}$$

2.2.4.3 Pearson Correlation Similarity

It uses Pearson Correlation Coefficient [47] to determine the similarity between users. The higher the coefficient the two users are more closely related. Formula to calculate Pearson Correlation Coefficient is given below,

$$r = \frac{\sum((u-\bar{u})(v-\bar{v}))}{\sqrt{\sum((u-\bar{u})^2 . \sum(v-\bar{v})^2}} \tag{2.4}$$

2.2.4.4 Mean Square Distance

The mean square distance is calculated as the ratio of the sum of the squares of the differences between the user-rated items and the user-rated items that are shared by both users. Then, by subtracting the mean square distance by one, the Mean Square similarity is calculated. Below is the formula for calculating the mean square distance.

$$sim(u,v)^{MSD} = 1 - \frac{\sum i \in I_{(u,v)}(R(u,i) - R(v,i))^2}{|I_{(u,v)}|} \tag{2.5}$$

R(u, i) and R(v, i) are the ratings given by the user u and v to item i, respectively. I(u, v) indicates the co-rated items of users u and v.

2.2.4.5 Jaccard Mean Square Distance (JMSD)

This is generated by multiplying two similarity measures, i.e. Jaccard similarity and mean square distance similarity measure [48].

$$sim(u,v)^{JMSD} = sim(u,v)^{Jaccard} . sim(u,v)^{MSD} \tag{2.6}$$

$$sim(u,v)^{JMSD} = \frac{|I_u \cap I_v|}{|I_u \cup I_v|} . (1 - \frac{\sum i \in I_{(u,v)}(R(u,i) - R(v,i))^2}{|I_{(u,v)}|}) \tag{2.7}$$

2.3 Conclusion

Recommender systems are a critical mode of information filtering in the digital era, where massive quantities of data are readily accessible. In this chapter, a detailed literature review is discussed. For convenience, different types of filtering techniques, Various uses, advantages, disadvantages are discussed. Thus, the aim of any recommender system is to construct a model in such a way that their consumer receives appropriate recommendations while maintaining the system's performance. The next chapter contains the detailed explanation of the proposed methodology of Movie recommendation system based on user preference, rating and comments framework.

2.3.1 Implementation Challenges

The main implementation problem was session over-problem in google colab,closed-environment, repetitive Task, Saving and Storage Problems. We were required to repeatedly perform the same set of actions to complete a task that was not only exhausting but also took a significant amount of time. It was challenging to keep all the packages running on google colab. As google colab does not provide persistent storage, We had to save the dataset after each model training. Again, to process this massive amount of data, high-speed hardware is required to run the models.

Chapter 3

Methodology

3.1 Introduction

Collaborative Filtering (CF) is an exciting technique for developing recommender systems. It provides users with personalized recommendations based on a database of their preferences, which identifies users with similar tastes. It then suggests items to a target user that other, similar users have enjoyed [49].

Additionally, user reviews can also be considered a subset of "user ratings," though they are typically expressed in natural language rather than numerical values. Although research on mining consumer preferences from feedback, also known as sentiment analysis or sentiment classification (e.g. [50], [51], [52], [53]), is gaining traction in the text mining literature, its integration with CF has received scant attention.

This paper describes my proposed framework for integrating sentiment analysis and CF. We employ a rating inference approach that derives a numerical score from textual reviews, allowing user preferences represented in the reviews to be easily fed into existing CF algorithms. This approach makes two contributions. First, it addresses the well-known data sparsity issue in CF by allowing CF algorithms to use textual reviews as an additional source of user preferences. Second, it enables the extension of CF to domains where numerical ratings for products are difficult to collect or where preferences for domain items are too complex to express as scalar ratings. We used the user-based CF algorithms and analyzed user reviews and inferring scores from them in this work.

3.2 Proposed Methodology

We used the user-based CF algorithms and analyzed user reviews and inferring scores from them in this work.

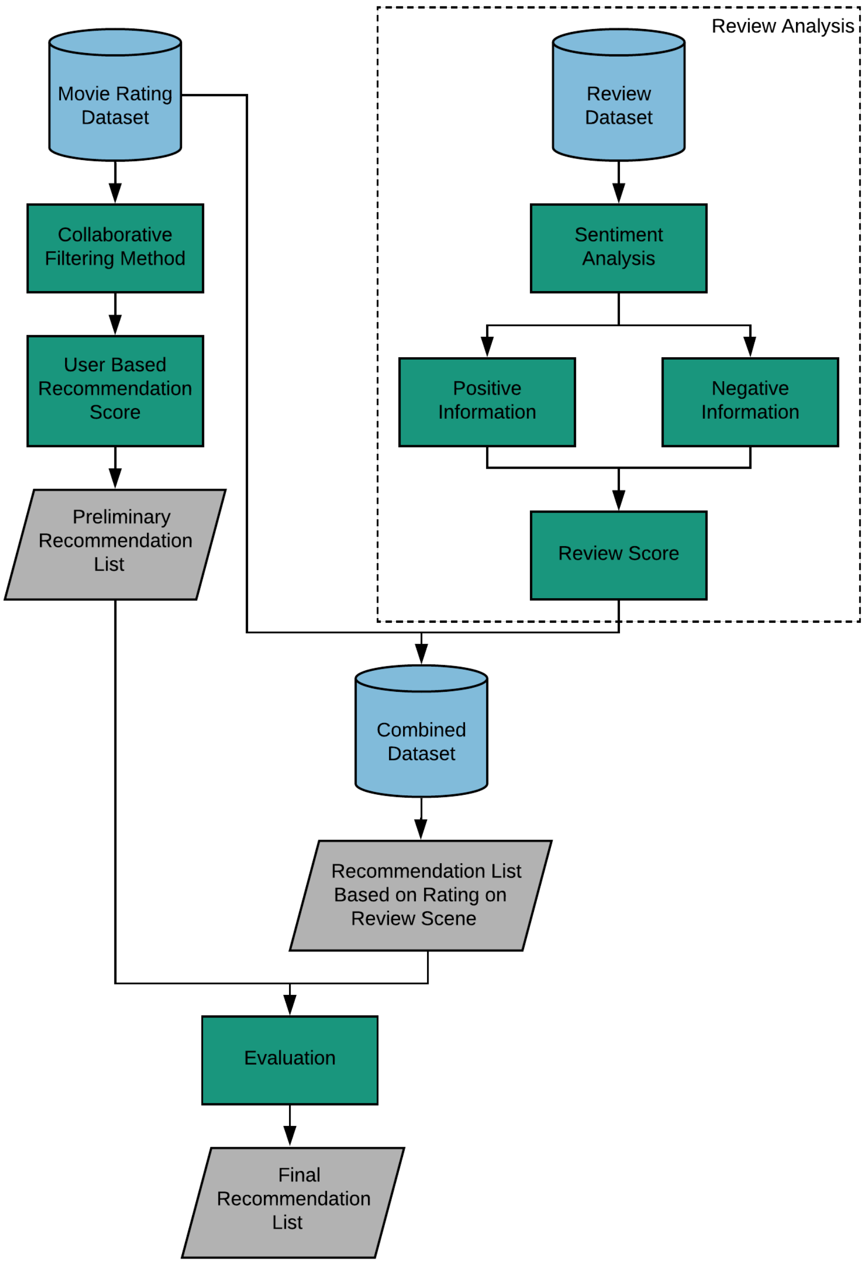

Review Analysis
Movie Rating Dataset
Review Dataset
Collaborative Filtering Method
Sentiment Analysis
User Based Recommendation Score
Positive Information
Negative Information
Preliminary Recommendation List
Review Score
Combined Dataset
Recommendation List Based on Rating on Review Scene
Evaluation
Final Recommendation List

Figure 3.1: Proposed Methodology

In figure 3.1 depicts the proposed system for movie recommendation. This section details the proposed recommender system's various steps and components.

3.3 Detailed Explanation

We calculated the review score using the positive and negative reviews generated by sentiment analysis.

Using a user-based collaborative filtering method, we generated a preliminary list from the rating dataset. Collaborative filtering is a technique that enlists the assistance of other users in recommending items to the input user. It seeks out users who share the input's preferences and opinions and then recommends things they've enjoyed to the information. There are numerous techniques for locating similar users (even some making use of Machine Learning). This one will be based on the Pearson Correlation Function. We calculated the weighted average recommendation score for similar users and recommended the items with the highest score.

We created a general recommendation list by combining the review score produced by sentiment analysis and the rating score generated by the preliminary rating list using the collaborative filtering process.

Finally, We produced the final recommendation list by combining and analyzing the preliminary rating list and the general recommended list we developed previously and presenting the top ten recommended movies.

My proposed model is a collaborative filtered recommender system whose results are augmented with the help of sentiment analysis scores. Experiments, both quantitative and qualitative, demonstrate the method's validity and effectiveness.

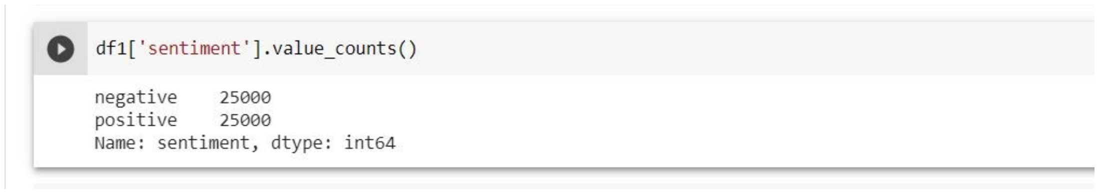
3.3.1 Sentiment Analysis Process

Sentiment analysis is a technique that researchers frequently employ to elicit people's opinions. In [54], sentiment analysis has been used to rate online software products. In this context, the research uses collaborative filtration algorithms and external reviews like sentimental analysis and subjective logic. The technique of sentimental analysis was employed in calculating the polarity and confidence of the sentences of review.

3.3.1.1 Implementation of Sentiment Analysis

The dataset is split into train and test data from the Rotten Tomatoes movie and critic reviews. My model then utilizes this information to aid in the analysis and visualization of the source data.

Following training, the model generates both positive and negative data from the review dataset.



```
df1['sentiment'].value_counts()

negative    25000
positive    25000
Name: sentiment, dtype: int64
```

Figure 3.2: sentiment analysis

In figure 3.2, shows a measure to evaluate the positive and negative prediction results, misclassification ratio was used, and by measuring the accuracy of the confusion matrix of Table 3.1 . Reviews with high performance were selected for the analysis

	Actual Positive	Actual Negetive
Predicted Positive	TF(True Positive)	FP(False Positive)
Predicted Negetive	FN(False Negetive)	TN(True Negetive)

Table 3.1: Confusion Matrix

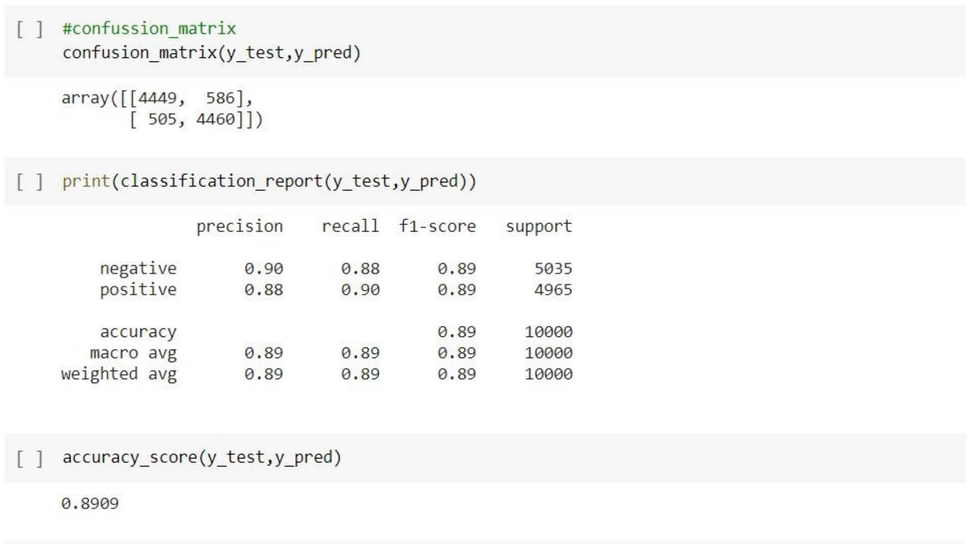
The TP(True Positive), FP(False Positive), TN(True Negative), and FN(False Negative) values in this matrix represent the result values. TP shows that the classifier is predicted to be a positive case accurately. On the other hand, FP indicates that the negative case was wrongly classed as positive by the classifier. Similarly, TN notes the classifier that the negative case is precisely predicted to be a negative case, while FN notes that the positive case was incorrectly classified as a negative by the classifier. Based on the results derived from the confusion matrix, accuracy, precision, and recall can be derived. Equations (3.1)–(3.4) respectively express the equations for calculating the accuracy, recall, precision, and F-measures.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

$$Precision = \frac{TP}{TP + FP} \tag{3.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3.3}$$

$$F - measure = \frac{2X(PrecisionXRecall}{Precision + Recall} \tag{3.4}$$



```
[ ]  #confussion_matrix
     confusion_matrix(y_test,y_pred)

     array([[4449,  586],
            [ 505, 4460]])

[ ]  print(classification_report(y_test,y_pred))

                   precision    recall  f1-score   support

         negative       0.90      0.88      0.89      5035
         positive       0.88      0.90      0.89      4965

         accuracy                           0.89     10000
        macro avg       0.89      0.89      0.89     10000
     weighted avg       0.89      0.89      0.89     10000


[ ]  accuracy_score(y_test,y_pred)

     0.8909
```

Figure 3.3: Performance measurement for sentiment analysis

In figure 3.3 shows the results of the accuracy calculation based on positive and negative reviews.

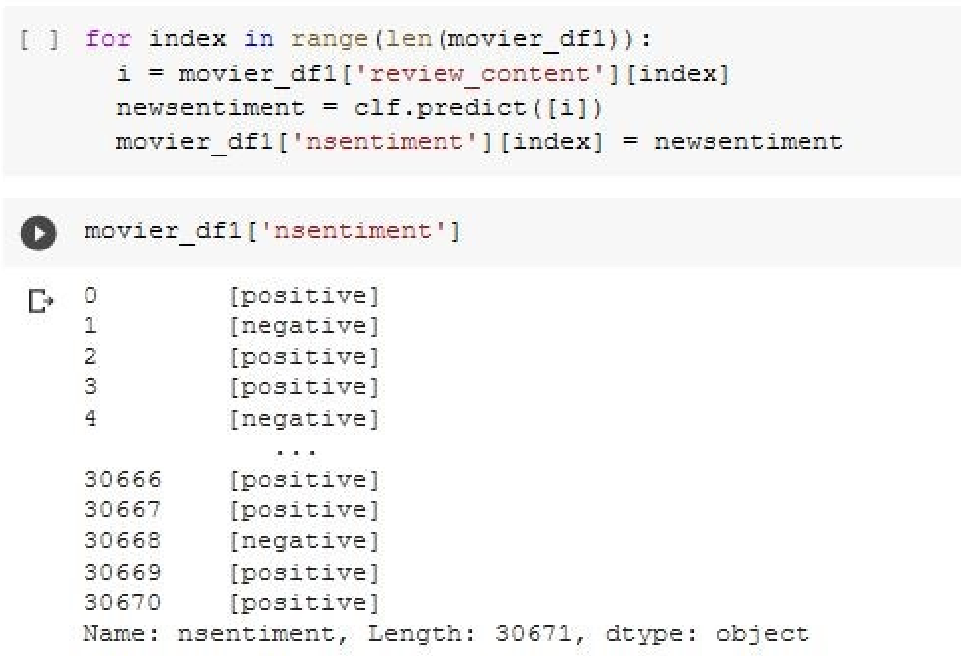
Additionally, this study utilized the SVM (support vector machine) (LinearSVC), a widely used technique for sentiment recognition and classification. Tokenization divides the raw text into words or sentences referred to as tokens. These tokens aid in comprehending the context or in developing the NLP model. The words must then be encoded as integers or floating-point values and fed into a machine learning algorithm called feature extraction (or vectorization). While word counts are a good starting point, they are extremely simplistic. One issue with simple counts is that some words, such as "the," will frequently appear in the encoded vectors, and their high counts will be meaningless. As an alternative, word frequencies can be calculated; by far, the most popular method is called TF-IDF. This is an acronym for "Term Frequency – Inverse Documentation."

The components of the resulting scores assigned to each word are frequency. The term frequency refers to the frequency with which a particular word appears in a document. Inverse Document Frequency: This algorithm reduces the frequency with which certain words appear in multiple documents. TF-IDF is word frequency scores that attempt to highlight more interesting words, e.g., those that are frequently used within a document but not across documents.

Here, The TfidfVectorizer tokenizes documents, learns the vocabulary, and calculates the inverse document frequency weightings, as well as allows us to encode new documents.

3.3.2 Positive Information and Negative Information

A list of positive and negative sentiments was generated following vectorization and tokenization.



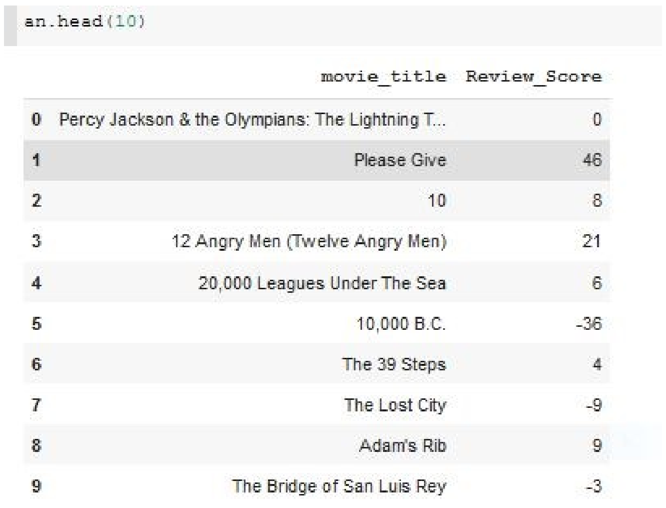
```
[ ] for index in range(len(movier_df1)):
      i = movier_df1['review_content'][index]
      newsentiment = clf.predict([i])
      movier_df1['nsentiment'][index] = newsentiment
```

```
movier_df1['nsentiment']
```

```
0        [positive]
1        [negative]
2        [positive]
3        [positive]
4        [negative]
            ...
30666    [positive]
30667    [positive]
30668    [negative]
30669    [positive]
30670    [positive]
Name: nsentiment, Length: 30671, dtype: object
```

Figure 3.4: Positive Information and Negative Information

In figure 3.4 the result is depicted. Following the sentiment analysis, we generated a review score, which we will use in the later stages of our recommendation system.



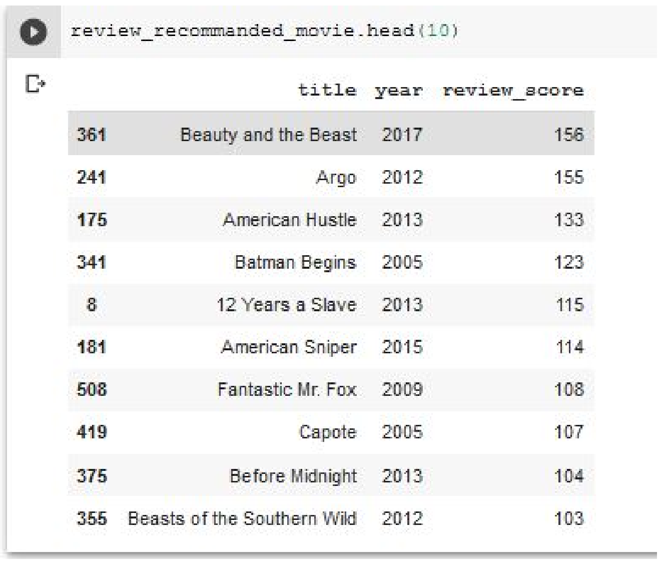
```
an.head(10)
```

	movie_title	Review_Score
0	Percy Jackson & the Olympians: The Lightning T...	0
1	Please Give	46
2	10	8
3	12 Angry Men (Twelve Angry Men)	21
4	20,000 Leagues Under The Sea	6
5	10,000 B.C.	-36
6	The 39 Steps	4
7	The Lost City	-9
8	Adam's Rib	9
9	The Bridge of San Luis Rey	-3

Figure 3.5: Review score of movies after sentiment analysis

In figure 3.5 shows the review score of movies after sentiment analysis. These findings imply that the constructed review score data is a more consistent and accurate sentiment analysis result.

3.3.3 Recommended List after Sentiment Analysis



```
review_recommanded_movie.head(10)
```

	title	year	review_score
361	Beauty and the Beast	2017	156
241	Argo	2012	155
175	American Hustle	2013	133
341	Batman Begins	2005	123
8	12 Years a Slave	2013	115
181	American Sniper	2015	114
508	Fantastic Mr. Fox	2009	108
419	Capote	2005	107
375	Before Midnight	2013	104
355	Beasts of the Southern Wild	2012	103

Figure 3.6: Recommended List after Sentiment Analysis

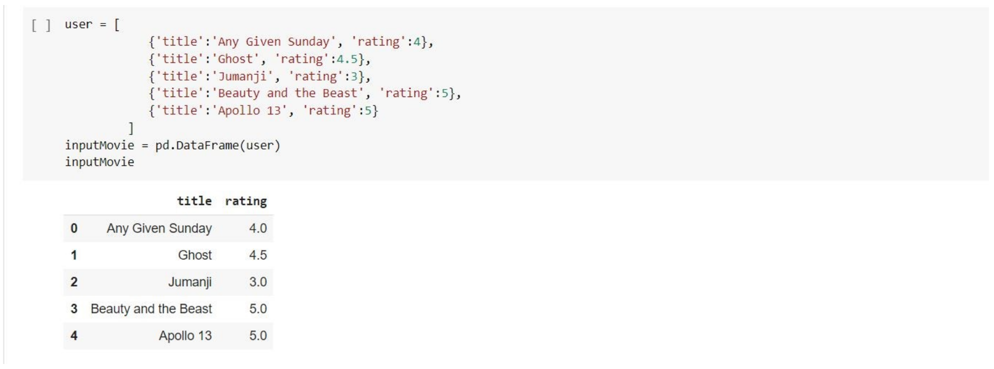
In figure 3.6 shows the recommended list after sentiment analysis.

3.3.4 Collaborative Filtering Method

Additionally referred to as User-User Filtering. It makes recommendations to the input user based on the actions of other users. It seeks out users who share the input's preferences and opinions and then recommends items they've enjoyed to the input. There are several methods for determining similar users (some of which incorporate machine learning), but the one I'll use here is based on the Pearson Correlation Function.

3.3.4.1 Implementation of Collaborative Filtering Method

In a fundamental scenario, collaborative filtering (CF) processing can be divided into four steps: step one is the collection of user rating data matrix, step two is the selection of similar neighbors based on rating similarity, and step three is Using a formula, determine the degree of similarity between two objects. And then Step four is to make recommendations for the best-rated items.

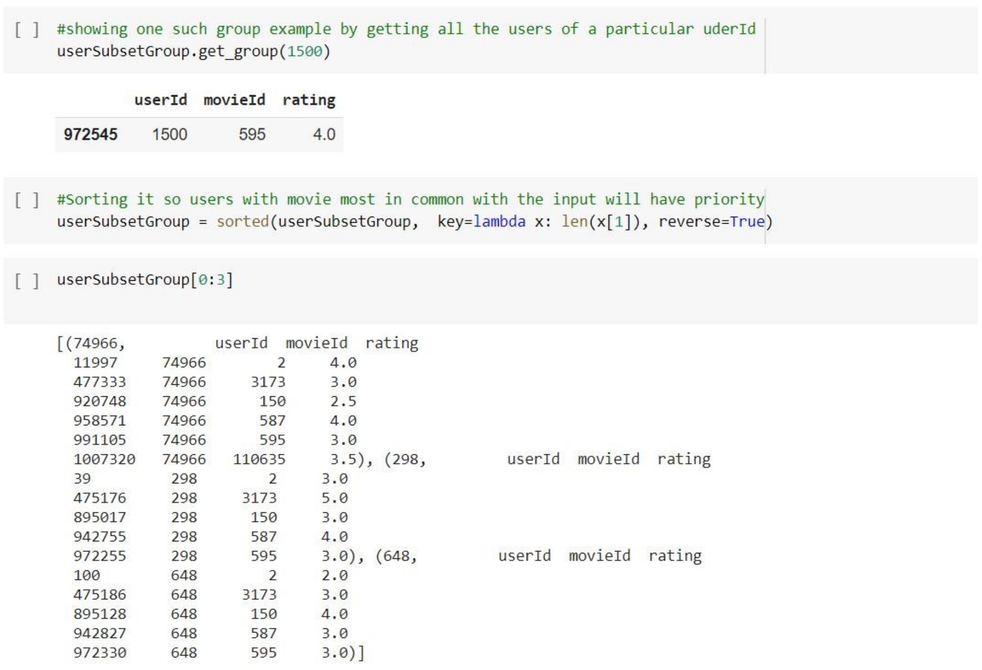


```
[ ] user = [
                {'title':'Any Given Sunday', 'rating':4},
                {'title':'Ghost', 'rating':4.5},
                {'title':'Jumanji', 'rating':3},
                {'title':'Beauty and the Beast', 'rating':5},
                {'title':'Apollo 13', 'rating':5}
           ]
    inputMovie = pd.DataFrame(user)
    inputMovie
```

	title	rating
0	Any Given Sunday	4.0
1	Ghost	4.5
2	Jumanji	3.0
3	Beauty and the Beast	5.0
4	Apollo 13	5.0

Figure 3.7: Collaborative filtering user based input data

In figure 3.7 demonstrates such group by obtaining all users with a particular userId. Using the rating data frame to identify users who have seen the same movies with the movie IDs in our input, we can now retrieve a subset of users who have viewed and reviewed the movie. Groupby creates multiple sub data-frames with identical values in the column specified as the parameter.



```
[ ] #showing one such group example by getting all the users of a particular uderId
    userSubsetGroup.get_group(1500)
```

```
        userId  movieId  rating
972545    1500      595     4.0
```

```
[ ] #Sorting it so users with movie most in common with the input will have priority
    userSubsetGroup = sorted(userSubsetGroup,  key=lambda x: len(x[1]), reverse=True)
```

```
[ ] userSubsetGroup[0:3]
```

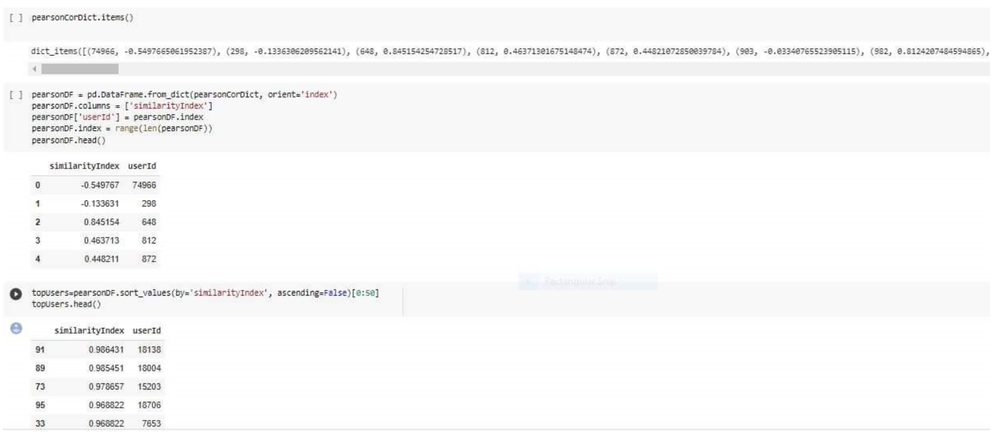
```
[(74966,          userId  movieId  rating
  11997     74966        2     4.0
  477333    74966     3173     3.0
  920748    74966      150     2.5
  958571    74966      587     4.0
  991105    74966      595     3.0
  1007320   74966   110635     3.5), (298,         userId  movieId  rating
  39          298        2     3.0
  475176      298     3173     5.0
  895017      298      150     3.0
  942755      298      587     4.0
  972255      298      595     3.0), (648,         userId  movieId  rating
  100         648        2     2.0
  475186      648     3173     3.0
  895128      648      150     4.0
  942827      648      587     3.0
  972330      648      595     3.0)]
```

Figure 3.8: User subset group- one user- different movies different rating

In figure 3.8 shows the user subset group of different movies, different ratings for one user. To find user similarity to the input user, We compared all users to our specified user to see which one is the most similar. We used the Pearson Correlation Coefficient to see how similar each user is to the input. It's a metric for assessing the strength of a linear relationship between two variables. The image below shows the formula for calculating this coefficient between sets X and Y with N values. The Pearson coefficient is unchanged when all elements are multiplied or added. So, for example, Pearson(X, 3) is equal to (3 + X). This property is vital in customer review systems because, for example, two users who rank items in differing degrees of importance might have differing opinions (and distinct taste).

$$r = \frac{\sum((u-\bar{u})(v-\bar{v}))}{\sqrt{\sum((u-\bar{u})^2 . \sum(v-\bar{v})^2}} \tag{3.5}$$

The formula returns values between r = -1 and r = 1, with 1 indicating a direct correlation between the two entities (perfect positive correlation) and -1 indicating a perfect negative correlation. A 1 indicates that the two users have similar tastes, whereas a -1 indicates the opposite.



```
pearsonCorDict.items()
```

```
dict_items([(74966, -0.5497665061952387), (298, -0.1336306209562141), (648, 0.845154254728517), (812, 0.46371301675148474), (872, 0.44821072850039784), (903, -0.03340765523905115), (982, 0.8124207484594865),
```

```
pearsonDF = pd.DataFrame.from_dict(pearsonCorDict, orient='index')
pearsonDF.columns = ['similarityIndex']
pearsonDF['userId'] = pearsonDF.index
pearsonDF.index = range(len(pearsonDF))
pearsonDF.head()
```

	similarityIndex	userId
0	-0.549767	74966
1	-0.133631	298
2	0.845154	648
3	0.463713	812
4	0.448211	872

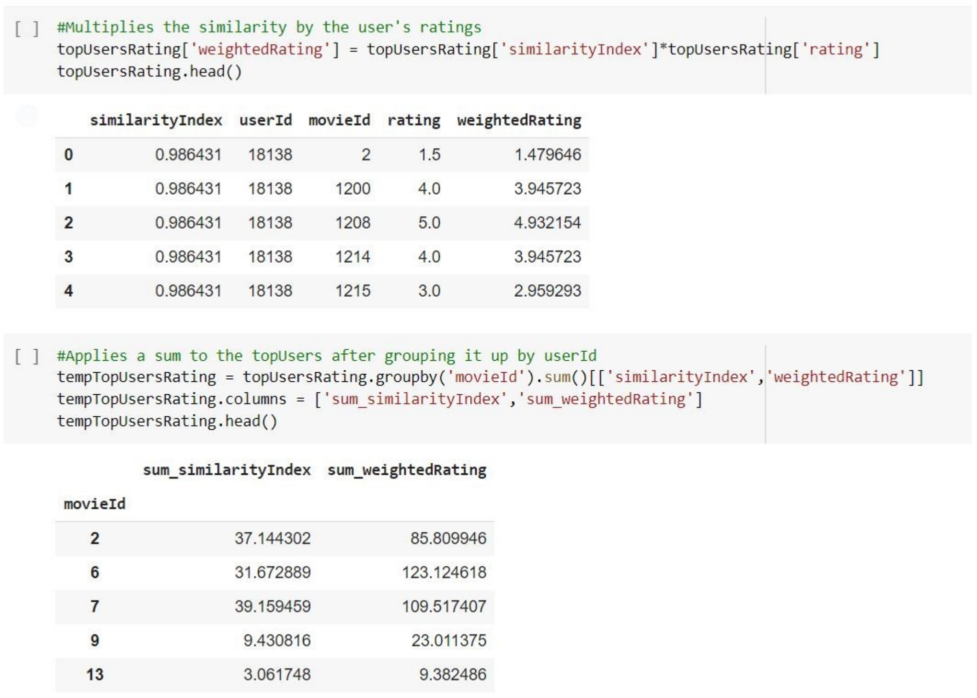
```
topUsers=pearsonDF.sort_values(by='similarityIndex', ascending=False)[0:50]
topUsers.head()
```

	similarityIndex	userId
91	0.986431	18138
89	0.985451	18004
73	0.978657	15203
95	0.968822	18706
33	0.968822	7653

Figure 3.9: Similarity Index of useId, (Pearson Correlation Coefficient)

3.3.4.2 Rating of selected users to all movies

We took the weighted average of the movie ratings using the Pearson Correlation as the weight to show the ratings of selected users to all movies. But, to do so, We first extracted the movies watched by the users in our pearsonDF from the rating data frame, then stored their correlation in a new column called similarityIndex. Then We just multiplied the movie rating by its weight (the similarity index), added the new ratings together, and divided by the sum of the weights. We did this by multiplying two columns, then grouping the dataframe by movieId, and finally dividing the dataframe into two columns. It compares the preferences of all similar users to potential movies for the input user.



```
#Multiplies the similarity by the user's ratings
topUsersRating['weightedRating'] = topUsersRating['similarityIndex']*topUsersRating['rating']
topUsersRating.head()
```

	similarityIndex	userId	movieId	rating	weightedRating
0	0.986431	18138	2	1.5	1.479646
1	0.986431	18138	1200	4.0	3.945723
2	0.986431	18138	1208	5.0	4.932154
3	0.986431	18138	1214	4.0	3.945723
4	0.986431	18138	1215	3.0	2.959293

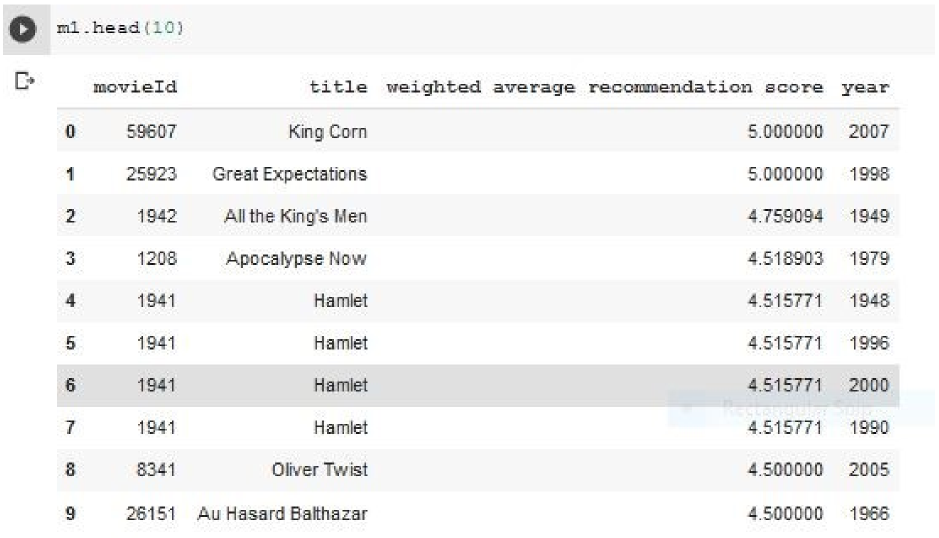
```
#Applies a sum to the topUsers after grouping it up by userId
tempTopUsersRating = topUsersRating.groupby('movieId').sum()[['similarityIndex','weightedRating']]
tempTopUsersRating.columns = ['sum_similarityIndex','sum_weightedRating']
tempTopUsersRating.head()
```

	sum_similarityIndex	sum_weightedRating
movieId		
2	37.144302	85.809946
6	31.672889	123.124618
7	39.159459	109.517407
9	9.430816	23.011375
13	3.061748	9.382486

Figure 3.10: topUsers after grouping it up by userId

3.3.4.3 Preliminary Recommendation List

Finally, using the collaborative filtering method, a preliminary list was generated.



```
m1.head(10)
```

	movieId	title	weighted average recommendation score	year
0	59607	King Corn	5.000000	2007
1	25923	Great Expectations	5.000000	1998
2	1942	All the King's Men	4.759094	1949
3	1208	Apocalypse Now	4.518903	1979
4	1941	Hamlet	4.515771	1948
5	1941	Hamlet	4.515771	1996
6	1941	Hamlet	4.515771	2000
7	1941	Hamlet	4.515771	1990
8	8341	Oliver Twist	4.500000	2005
9	26151	Au Hasard Balthazar	4.500000	1966

Figure 3.11: Preliminary Recommendation List

In figure 3.11 illustrates the outcome of collaborative filtering method.

3.4 Conclusion

We proposed a framework for developing a movie recommendation system that takes user preference, movie ratings, and user comments into account. A rating inference approach is used to integrate sentiment analysis and CF in this case. This approach converts unstructured natural language texts expressing user preferences into numerical scales understandable by existing CF algorithms. The proposed system for movie recommendation Considers other users' ratings and adapts to the user's changing interests. This study proposed a new recommendation method that enhances the performance of collaborative filtering by incorporating qualitative data, namely user reviews.

Chapter 4

Results and Discussions

4.1 Introduction

The preceding chapter provided an in-depth explanation of the proposed framework for Designing a Movie Recommendation System Using User Preferences, Movie Ratings, and Comments. This chapter evaluates the proposed framework's performance. The Rotten Tomatoes movies and critic reviews dataset, as well as the MovieLens 100k rating dataset, were used in this thesis. This dataset was obtained through kaggle [1], an open-source data repository.

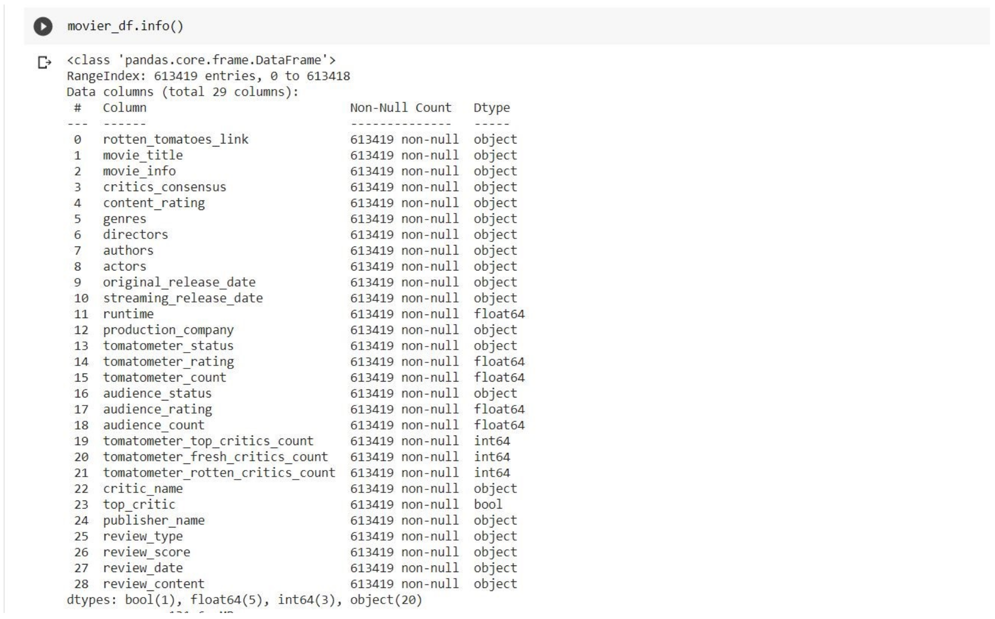
4.2 Dataset Description

The movie review dataset for this research was obtained from the Rotten Tomatoes movies and critic reviews dataset [2], and the rating dataset was obtained from the MovieLens 100k dataset [3]. The Rotten tomatoes and critical reviews data set consists of 28 feature data columns of 600k data.

[1]https://www.kaggle.com

[2]https://www.kaggle.com/stefanoleone992/rotten-tomatoes-movies-and-critic-reviews-dataset

[3]https://www.kaggle.com/grouplens/movielens-20m-dataset

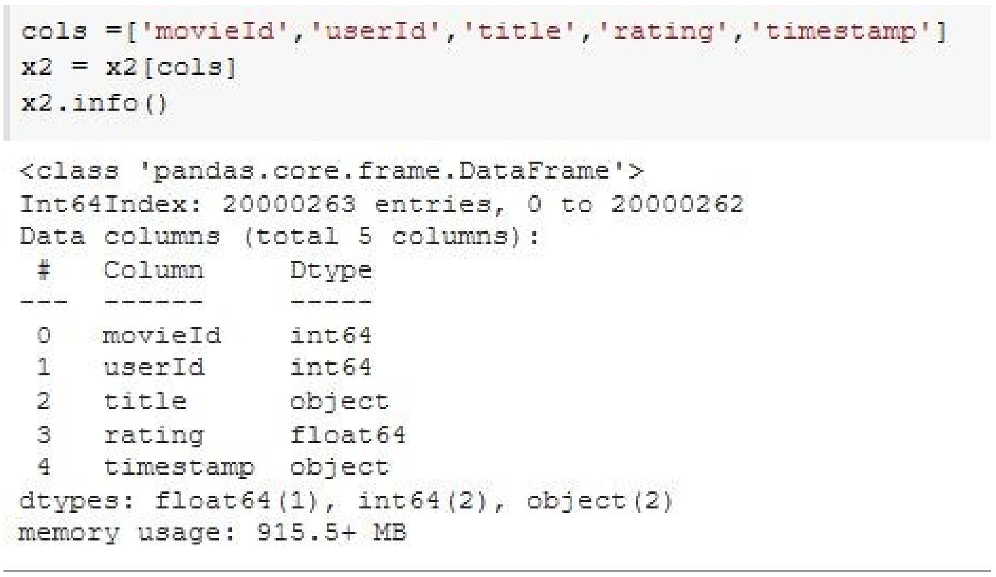


```
movier_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 613419 entries, 0 to 613418
Data columns (total 29 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   rotten_tomatoes_link              613419 non-null  object 
 1   movie_title                       613419 non-null  object 
 2   movie_info                        613419 non-null  object 
 3   critics_consensus                 613419 non-null  object 
 4   content_rating                    613419 non-null  object 
 5   genres                            613419 non-null  object 
 6   directors                         613419 non-null  object 
 7   authors                           613419 non-null  object 
 8   actors                            613419 non-null  object 
 9   original_release_date             613419 non-null  object 
 10  streaming_release_date            613419 non-null  object 
 11  runtime                           613419 non-null  float64
 12  production_company                613419 non-null  object 
 13  tomatometer_status                613419 non-null  object 
 14  tomatometer_rating                613419 non-null  float64
 15  tomatometer_count                 613419 non-null  float64
 16  audience_status                   613419 non-null  object 
 17  audience_rating                   613419 non-null  float64
 18  audience_count                    613419 non-null  float64
 19  tomatometer_top_critics_count     613419 non-null  int64  
 20  tomatometer_fresh_critics_count   613419 non-null  int64  
 21  tomatometer_rotten_critics_count  613419 non-null  int64  
 22  critic_name                       613419 non-null  object 
 23  top_critic                        613419 non-null  bool   
 24  publisher_name                    613419 non-null  object 
 25  review_type                       613419 non-null  object 
 26  review_score                      613419 non-null  object 
 27  review_date                       613419 non-null  object 
 28  review_content                    613419 non-null  object 
dtypes: bool(1), float64(5), int64(3), object(20)
```

Figure 4.1: Review Dataset Information

In figure 4.1 shows the review dataset.



```
cols =['movieId','userId','title','rating','timestamp']
x2 = x2[cols]
x2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000263 entries, 0 to 20000262
Data columns (total 5 columns):
 #   Column     Dtype  
---  ------     -----  
 0   movieId    int64  
 1   userId     int64  
 2   title      object 
 3   rating     float64
 4   timestamp  object 
dtypes: float64(1), int64(2), object(2)
memory usage: 915.5+ MB
```

Figure 4.2: Rating Dataset Information

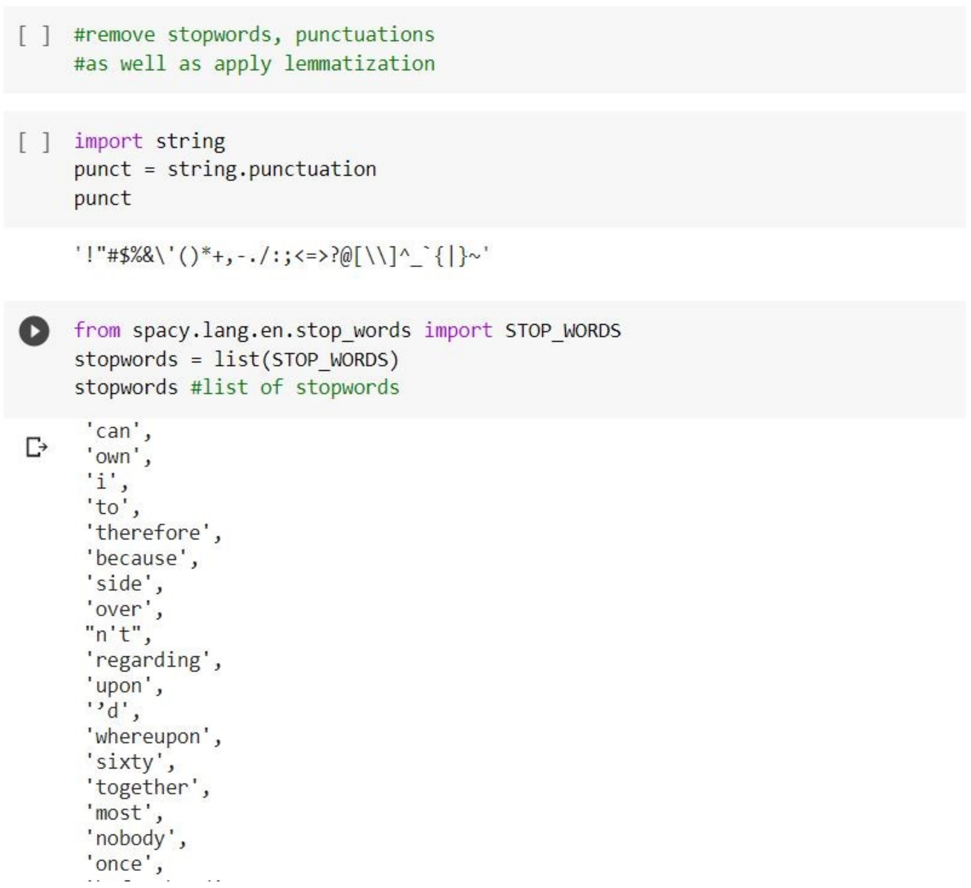
In figure 4.2, The rating dataset consists of five movie columns containing a total of twenty million data.

4.2.1 Dataset Preprocessing

We've removed duplicate or non-identifiable movies and crawled Rotten Tomatoes movies and critical reviews for the rest of our movies.

4.2.1.1 Dataset Cleaning

For dataset cleaning, We have removed duplicate or non-identifiable movies and crawled Rotten Tomatoes movies and critical reviews for the rest of our movies. We have also cleaned the rating data set.



```
[ ] #remove stopwords, punctuations
    #as well as apply lemmatization
```

```
[ ] import string
    punct = string.punctuation
    punct

    '!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

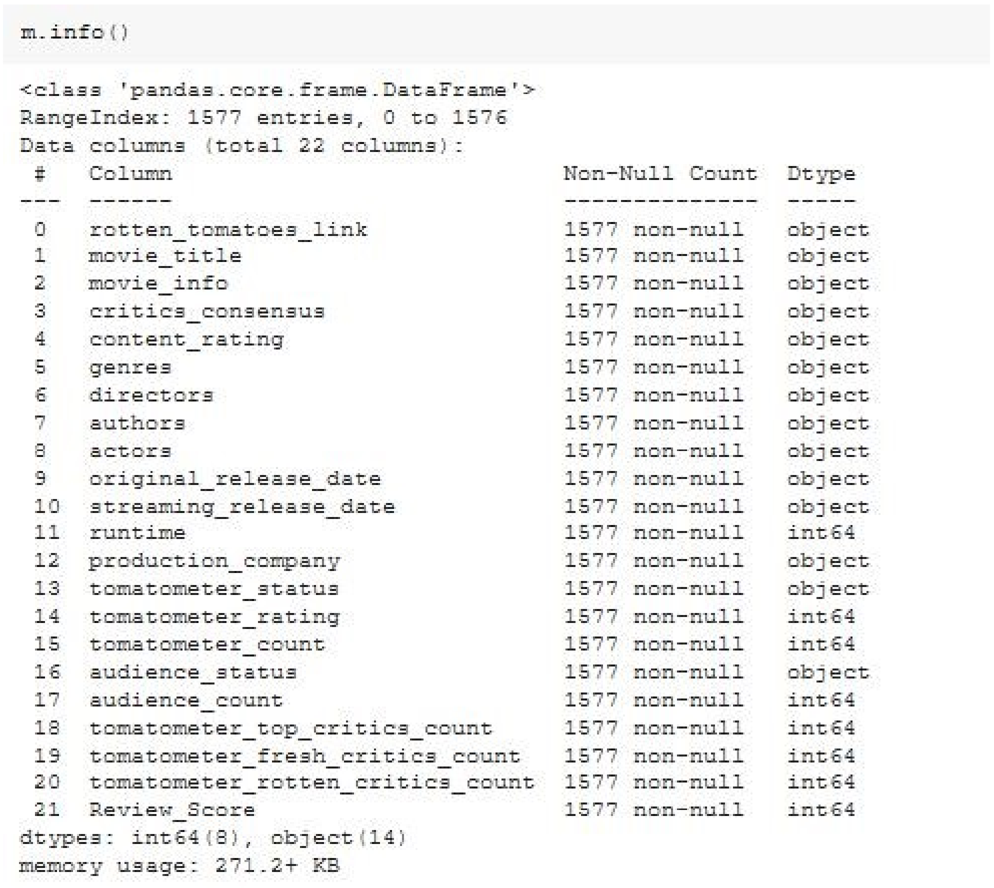
```
from spacy.lang.en.stop_words import STOP_WORDS
stopwords = list(STOP_WORDS)
stopwords #list of stopwords

 'can',
 'own',
 'i',
 'to',
 'therefore',
 'because',
 'side',
 'over',
 "n't",
 'regarding',
 'upon',
 ''d',
 'whereupon',
 'sixty',
 'together',
 'most',
 'nobody',
 'once',
```

Figure 4.3: Removing stop-words and punctuation from review content.

In this figure 4.3 the attribute ’review_content’ from ‘Rotten Tomatoes movies and critic reviews dataset’ is tokenized and cleaned for sentiment analysis.

4.2.1.2 Modified Dataset

For modifying the ‘Rotten Tomatoes movies and critic reviews dataset,’ We have analyzed the sentiment divided the ‘review_content’ attribute into two classes: ‘positive’ and ‘negative’. Then the ‘positive’ review valued as ‘+1’ and the ‘negative’ review valued as ‘-1’, and a new attribute, ‘review_content’ is formed. Then We evaluated the ‘review_content’ for each movie using the ‘movie_title’.



```
m.info()
```

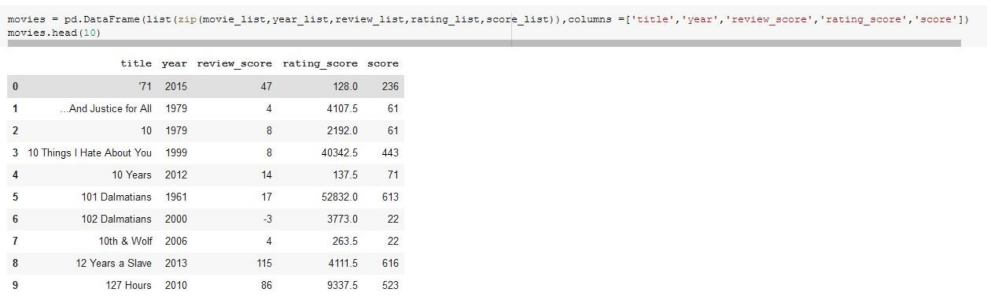
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1577 entries, 0 to 1576
Data columns (total 22 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   rotten_tomatoes_link              1577 non-null   object
 1   movie_title                       1577 non-null   object
 2   movie_info                        1577 non-null   object
 3   critics_consensus                 1577 non-null   object
 4   content_rating                    1577 non-null   object
 5   genres                            1577 non-null   object
 6   directors                         1577 non-null   object
 7   authors                           1577 non-null   object
 8   actors                            1577 non-null   object
 9   original_release_date             1577 non-null   object
 10  streaming_release_date            1577 non-null   object
 11  runtime                           1577 non-null   int64
 12  production_company                1577 non-null   object
 13  tomatometer_status                1577 non-null   object
 14  tomatometer_rating                1577 non-null   int64
 15  tomatometer_count                 1577 non-null   int64
 16  audience_status                   1577 non-null   object
 17  audience_count                    1577 non-null   int64
 18  tomatometer_top_critics_count     1577 non-null   int64
 19  tomatometer_fresh_critics_count   1577 non-null   int64
 20  tomatometer_rotten_critics_count  1577 non-null   int64
 21  Review_Score                      1577 non-null   int64
dtypes: int64(8), object(14)
memory usage: 271.2+ KB
```

Figure 4.4: Modified movie dataset for 'Rotten Tomatoes movies and critic reviews dataset'.

In figure 4.4 shows that the 'Rotten Tomatoes movies and critic reviews dataset' has 1577 data as the dataset generated the 'Review_Score' according to the movie and thus it reduced the redundant data.

4.2.1.3 Dataset Merging

After modified the 'Rotten Tomatoes movies and critic reviews dataset' is merged with the 'rating dataset' for generating a recommendation system based on review score and rating score.



```
movies = pd.DataFrame(list(zip(movie_list,year_list,review_list,rating_list,score_list)),columns =['title','year','review_score','rating_score','score'])
movies.head(10)
```

	title	year	review_score	rating_score	score
0	'71	2015	47	128.0	236
1	...And Justice for All	1979	4	4107.5	61
2	10	1979	8	2192.0	61
3	10 Things I Hate About You	1999	8	40342.5	443
4	10 Years	2012	14	137.5	71
5	101 Dalmatians	1961	17	52832.0	613
6	102 Dalmatians	2000	-3	3773.0	22
7	10th & Wolf	2006	4	263.5	22
8	12 Years a Slave	2013	115	4111.5	616
9	127 Hours	2010	86	9337.5	523

Figure 4.5: Merged movie dataset

In figure 4.5, shows that the dataset ‘Rotten Tomatoes movies and critic reviews dataset’ is merged with the ‘rating dataset’ and thus formed a movie data set which has ‘title’, ‘year’, ‘review_score’,’rating_score’ and ‘score’ as attributes. Here, ‘score’ attribute is generated from ‘review_score’ and ‘rating_score’.

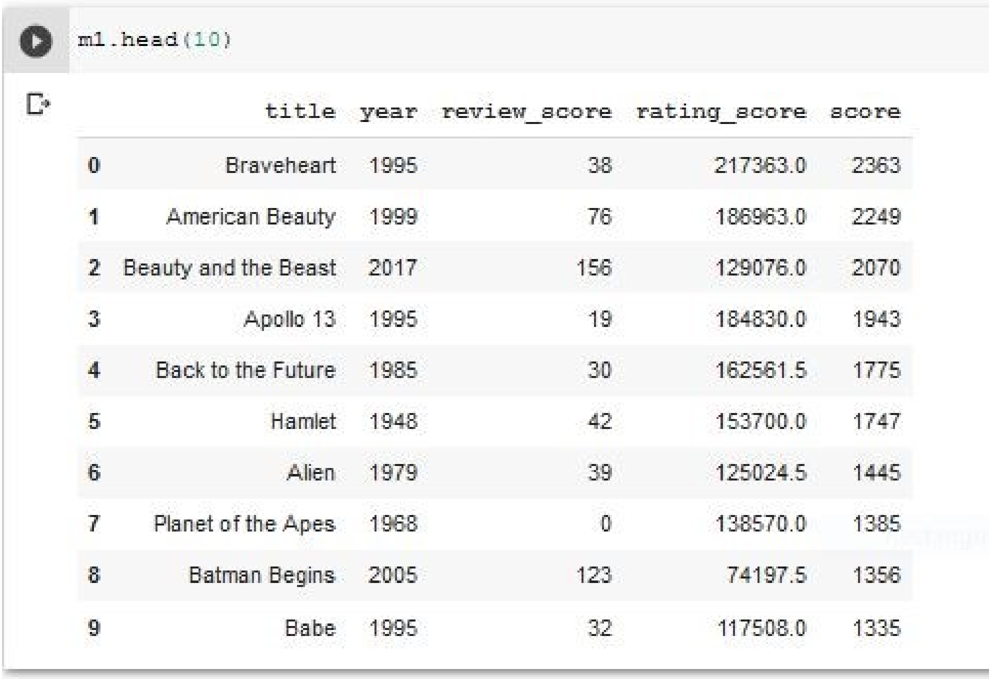
4.3 Evaluation of Framework

This thesis work begins with Rotten Tomatoes movies and critic reviews dataset and the MovieLens 100k dataset that are pre-processed for subsequent computations. The proposed framework’s optimal features are the rating score, review score, users, and movie title.

We created the preliminary list using the collaborative filtering method based on the weighted average recommendation score. Then, We analyzed the review content to derive sentiment, which can be classified as "negative" or "positive" information. The system assigned a score of (-1) to "negative" sentiment and (+1) to "positive" sentiment. The total review score is calculated by adding the scores for each movie. Based on this review and rating score, the system generated a recommendation list. Then We evaluated the two recommendation lists to create the proposed system, which generated the final list of recommended movies.

4.3.1 Recommendation List based on rating and review

The combined data set of the movie dataset and the review score gives us the recommendation list.



```
m1.head(10)
```

```
               title  year  review_score  rating_score  score
0         Braveheart  1995            38      217363.0   2363
1    American Beauty  1999            76      186963.0   2249
2  Beauty and the Beast  2017        156      129076.0   2070
3          Apollo 13  1995            19      184830.0   1943
4  Back to the Future  1985            30      162561.5   1775
5             Hamlet  1948            42      153700.0   1747
6              Alien  1979            39      125024.5   1445
7  Planet of the Apes  1968             0      138570.0   1385
8      Batman Begins  2005           123       74197.5   1356
9               Babe  1995            32      117508.0   1335
```

Figure 4.6: Recommendation List based on rating and review

In figure 4.6 shows the recommendation list based on rating and review.

4.4 Evaluation of Performance

We use statistical accuracy metrics to determine the accuracy of a recommendation system. The average absolute error (MAE) and root-mean-square error (RMSE) are utilized in the evaluation method to assess the difference between a recommended system method that only reflects rating data and the method that integrates the rating data with sentiment values. N is the total number of actual ratings in an item set in the following formula.

$$MAE = \frac{\sum |R_{i,j} - \bar{R}_{i,j}|}{N} \tag{4.1}$$

The lower the MAE value, the more accurate the recommendation system's prediction of user ratings is.

RMSE is a measure of RMSE that first gets the total of squared values differences and then divides the sum result by the number of predictions and then the square root. RMSE is a measure of RMSE. Smaller error values indicate a better preview of the recommendation system in both of these measures.

$$RMSE = \sqrt{\frac{\sum(R_{i,j} - \bar{R}_{i,j})^2}{N}} \tag{4.2}$$

In this section, N is the number of data points; Rij indicates the actual item j rating provided by user I and ÅRij indicates the user rating prediction. MAE is a medium absolute error measure calculated by adding all the absolute error values from the value measured to the value predicted and divided by the number of values predicted.

The MAE and RMSE values were used to compare the performance of the prediction method proposed in this paper. Each movie has a unique rating based on its audience. We compared each user's movie rating to the average. 80% of the data was used for training purposes, while 20% was used for testing purposes.

Additionally, cross-validation was used to assess the performance of the rating prediction algorithm. The experimental results of a five fold cross-validation procedure are depicted in the following table:

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
RMSE	0.9637	0.8849	1.5607	1.2057	1.1256
MAE	0.8442	0.7376	1.2425	1.1800	0.9038

Table 4.1: Performance Measurement of Preliminary List using Collaborative Filtering Method

In table 4.1 shows the performance measurement of the system. It operates solely on the basis of a given movie ID and attempts to predict ratings based on how other users have predicted the movie.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
RMSE	0.9637	0.8849	1.5607	1.2057	1.1256
MAE	0.8442	0.7376	1.2425	1.1800	0.9038

Table 4.2: Performance Measurement of Recommendation List after Sentiment Analysis

In table 4.2, it works purely on the basis of an assigned movie ID and tries to predict review_score based on how the other users have predicted the movie.

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
RMSE	0.2893	0.1603	0.3190	0.2528	0.2598
MAE	0.2514	0.1369	0.3190	0.2210	0.2301

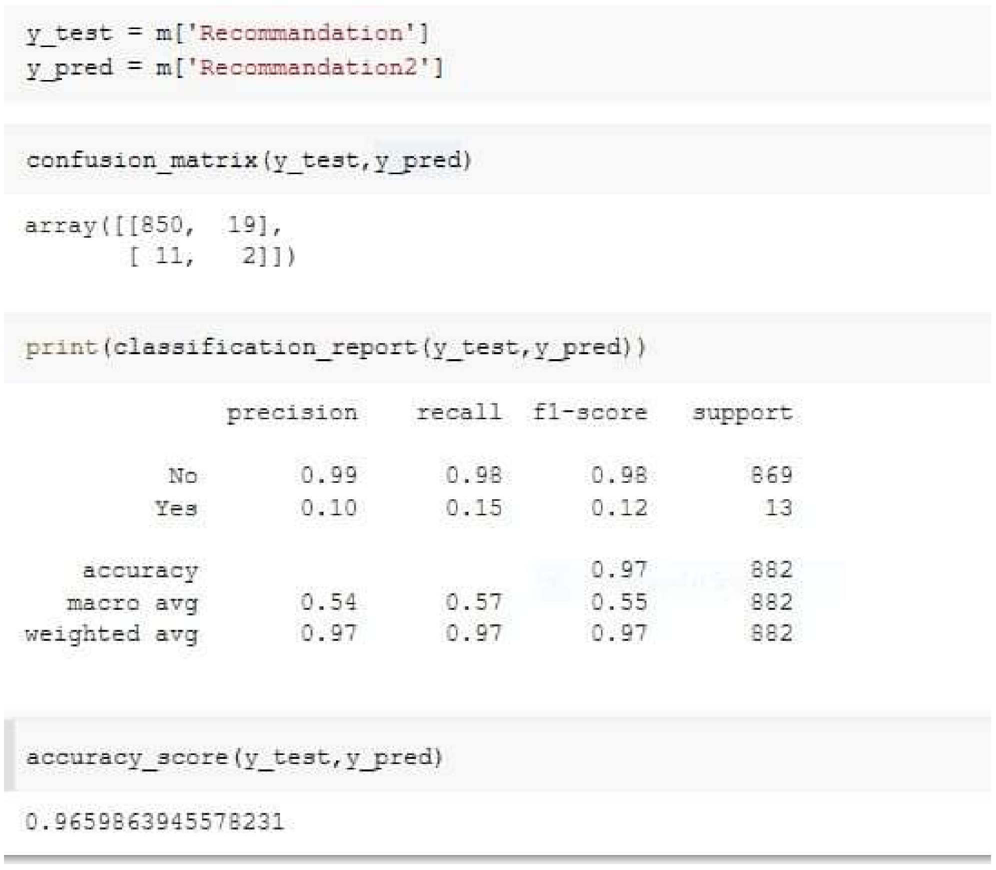
Table 4.3: Performance Measurement of Final Recommendation List

In table 4.3, it works purely on the basis of an assigned movie and tries to predict combined rating and review score based on how the other users have predicted the movie on the preliminary list.

Evaluation	FINAL RECOMMENDATION LIST	PRELIMINARY LIST BASED ON CF FILTERING	RECOMMENDATION LIST BASED ON Sentiment anlaysis
RMSE & MAE	The RMSE and MAE values are the smallest of all of them.	Due to the fact that this list is based on Collaborative Filtering, its value is lower than the list based on sentiment analysis.	The RMSE and MAE are the largest, indicating that this method is not very accurate.
FIT AND TEST TIME	As the RMSE and MAE values are small, the method requires the least time (nearly 0.0) to fit and test the model.	It will take slightly longer than the final recommended list to fit and test the model.	It takes a long time to fit and test the model.

Table 4.4: Comparison between Performance Measurements

In table 4.4, as can be seen, the final recommendation list has a lower RMSE and MAE value than the sentiment-analyzed recommended list, as well as a lower RMSE and MAE value than the preliminary recommended list.



```
y_test = m['Recommandation']
y_pred = m['Recommandation2']
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[850,  19],
       [ 11,   2]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

          No       0.99      0.98      0.98       869
         Yes       0.10      0.15      0.12        13

    accuracy                           0.97       882
   macro avg       0.54      0.57      0.55       882
weighted avg       0.97      0.97      0.97       882
```

```
accuracy_score(y_test,y_pred)
```

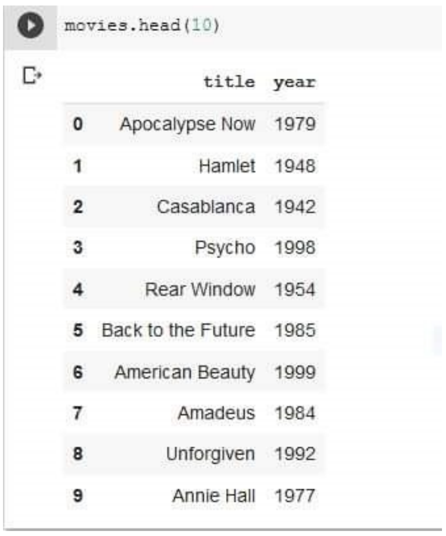
```
0.9659863945578231
```

Figure 4.7: Accuracy score final recommendation list

In figure 4.7 shows the accuracy score of final recommendation list. Here, y_test data represents the preliminary movie list generated using the collaborative filtering method, while y_pred data represents the recommendation list generated using the review and rating score generated previously. Accuracy (ACC) is calculated by dividing the total number of correct predictions by the total number of observations in the dataset. The highest level of accuracy is 1.0, while the lowest level is 0.0. As illustrated in the figure above, the model's accuracy is 0.965986. In this method, our established technique achieves a detection rate of 96.86%.

4.4.1 Final Recommendation List

Finally, we are able to provide a final recommendation list for a total of K movies.



```
movies.head(10)
```

```
               title  year
0     Apocalypse Now  1979
1             Hamlet  1948
2         Casablanca  1942
3             Psycho  1998
4        Rear Window  1954
5 Back to the Future  1985
6    American Beauty  1999
7            Amadeus  1984
8         Unforgiven  1992
9         Annie Hall  1977
```

Figure 4.8: Final Recommendation List

In figure 4.8, we see the proposed system's desired recommendation list.

4.5 Conclusion

The results of the proposed framework for A Framework for Designing a Movie Recommendation System Based on User Preferences, Movie Ratings, and Comments are presented in this chapter. The performance metric is quite nicely illustrated. The following chapter summarizes this thesis and makes recommendations for the future.

Chapter 5

Conclusion

5.1 Conclusion

To improve the accuracy of the existing collaborative filtering method, which generates recommendation results solely based on qualitative data, this study proposed a new recommendation method that incorporates qualitative data, namely user reviews, into the collaborative filtering performance. The findings of this thesis indicate that the framework we developed using sentiment analysis in conjunction with collaborative filtering produces more stable and accurate results. A detailed description of the data modification and merging process is provided, along with an assessment of its quality. In general, the SVM weighting technique, which generates a weighted average recommendation score, resulted in the most stable and effective performance improvement.

The proposed method for recommendation systems in this thesis is expected to reflect user preferences in recommendation systems accurately. The technique described in this thesis quantifies user review data while addressing the limitations of previous studies that determined user preferences solely from rating data. The findings of this study can be applied to new studies and to my further development of the corpus. As previously stated, our rating inference approach converts textual reviews to ratings, making sentiment analysis and CF easy to integrate. The experimental results demonstrate that this approach has a significant impact on prediction accuracy and execution time when compared to traditional UBCF and IBCF. It results in an improvement of the recommendation system's quality through the use of collaborative filtering.

5.2 Future Work

We are aware of the possibility of performing text-based CF directly from a collection of user reviews, however. A possible solution is to model text-based CF as an information retrieval (IR) problem, with reviews written by the target user serving as the "query" and those written by other similar users serving as the "relevant documents," from which the target user's recommendations can be generated. This continues to be an intriguing area of research for future research. In the future, studies utilizing the proposed recommendation system technique should be conducted to develop a variety of sentiment-based recommendation systems. We can further refine the sentiment scores and increase the recommendation system's accuracy.

References

[1] J. Bobadilla, F. Ortega, A. Hernando and A. Gutiérrez, 'Recommender systems survey,' *Knowledge-based systems*, vol. 46, pp. 109–132, 2013 (cit. on p. 3).

[2] M. Sharma and S. Mann, 'A survey of recommender systems: Approaches and limitations,' *International Journal of Innovations in Engineering and Technology*, vol. 2, no. 2, pp. 8–14, 2013 (cit. on p. 3).

[3] F. Mansur, V. Patel and M. Patel, 'A review on recommender systems,' in *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, IEEE, 2017, pp. 1–6 (cit. on p. 4).

[4] P. V. Kumar and V. R. Reddy, 'A survey on recommender systems (rss) and its applications,' *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 2, no. 8, pp. 5254–5260, 2014 (cit. on p. 4).

[5] C. A. Gomez-Uribe and N. Hunt, 'The netflix recommender system_algorithms,' *Bus. Value Pdf*, vol. 6, no. 4, 2015 (cit. on p. 4).

[6] G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie and Z. Li, 'Drn: A deep reinforcement learning framework for news recommendation,' in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 167–176 (cit. on p. 4).

[7] W.-C. Kang, C. Fang, Z. Wang and J. McAuley, 'Visually-aware fashion recommendation and design with generative image models,' in *2017 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2017, pp. 207–216 (cit. on p. 4).

[8] V. C. Heung, H. Qu and R. Chu, 'The relationship between vacation factors and socio-demographic and travelling characteristics: The case of japanese leisure travellers,' *Tourism management*, vol. 22, no. 3, pp. 259–269, 2001 (cit. on p. 5).

[9] J. A. Konstan and J. Riedl, 'Recommender systems: From algorithms to user experience,' *User modeling and user-adapted interaction*, vol. 22, no. 1, pp. 101–123, 2012 (cit. on p. 5).

[10] C. Pan and W. Li, 'Research paper recommendation with topic analysis,' in *2010 International Conference On Computer Design and Applications*, IEEE, vol. 4, 2010, pp. V4–264 (cit. on p. 5).

[11] P. Pu, L. Chen and R. Hu, 'A user-centric evaluation framework for recommender systems,' in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 157–164 (cit. on pp. 5, 8).

[12] B. Pathak, R. Garfinkel, R. D. Gopal, R. Venkatesan and F. Yin, 'Empirical analysis of the impact of recommender systems on sales,' *Journal of Management Information Systems*, vol. 27, no. 2, pp. 159–188, 2010 (cit. on pp. 5, 8).

[13] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan and J. Riedl, 'Getting to know you: Learning new user preferences in recommender systems,' in *Proceedings of the 7th international conference on Intelligent user interfaces*, 2002, pp. 127–134 (cit. on p. 9).

[14] J. B. Schafer, J. Konstan and J. Riedl, 'Recommender systems in e-commerce,' in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 158–166 (cit. on p. 9).

[15] P. Resnick and H. R. Varian, 'Recommender systems,' *Communications of the ACM*, vol. 40, no. 3, pp. 56–58, 1997 (cit. on p. 9).

[16] A. M. Acilar and A. Arslan, 'A collaborative filtering method based on artificial immune network,' *Expert Systems with Applications*, vol. 36, no. 4, pp. 8324–8332, 2009 (cit. on p. 9).

[17] L.-S. Chen, F.-H. Hsu, M.-C. Chen and Y.-C. Hsu, 'Developing recommender systems with the consideration of product profitability for sellers,' *Information Sciences*, vol. 178, no. 4, pp. 1032–1048, 2008 (cit. on p. 9).

[18] M. Jalali, N. Mustapha, M. N. Sulaiman and A. Mamat, 'Webpum: A web-based recommendation system to predict user future movements,' *Expert Systems with Applications*, vol. 37, no. 9, pp. 6201–6212, 2010 (cit. on p. 9).

[19] G. Adomavicius and A. Tuzhilin, 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,' *IEEE transactions on knowledge and data engineering*, vol. 17, no. 6, pp. 734–749, 2005 (cit. on p. 9).

[20] C.-N. Ziegler, S. M. McNee, J. A. Konstan and G. Lausen, 'Improving recommendation lists through topic diversification,' in *Proceedings of the 14th international conference on World Wide Web*, 2005, pp. 22–32 (cit. on p. 9).

[21] S.-H. Min and I. Han, 'Detection of the customer time-variant pattern for improving recommender systems,' *Expert Systems with Applications*, vol. 28, no. 2, pp. 189–199, 2005 (cit. on p. 10).

[22] Ò. Celma and X. Serra, 'Foafing the music: Bridging the semantic gap in music recommendation,' *Journal of Web Semantics*, vol. 6, no. 4, pp. 250–256, 2008 (cit. on p. 10).

[23] H. Lieberman *et al.*, 'Letizia: An agent that assists web browsing,' *IJCAI (1)*, vol. 1995, pp. 924–929, 1995 (cit. on p. 10).

[24] M. J. Pazzani, ‘A framework for collaborative, content-based and demographic filtering,’ *Artificial intelligence review*, vol. 13, no. 5, pp. 393–408, 1999 (cit. on pp. 10, 12).

[25] A. Jennings and H. Higuchi, ‘A personal news service based on a user model neural network,’ *IEICE Transactions on Information and Systems*, vol. 75, no. 2, pp. 198–209, 1992 (cit. on pp. 10, 12).

[26] M. Göksedef and Ş. Gündüz-Öğüdücü, ‘Combination of web page recommender systems,’ *Expert Systems with Applications*, vol. 37, no. 4, pp. 2911–2922, 2010 (cit. on pp. 10, 13).

[27] B. Mobasher, *Recommender systems. kunstliche intelligenz. special issue on web mining*, 2007 (cit. on p. 10).

[28] M. Y. H. Al-Shamri and K. K. Bharadwaj, ‘Fuzzy-genetic approach to recommender systems based on a novel hybrid user model,’ *Expert systems with applications*, vol. 35, no. 3, pp. 1386–1399, 2008 (cit. on pp. 10, 13).

[29] D. Mican and N. Tomai, ‘Association-rules-based recommender system for personalization in adaptive web-based applications,’ in *International Conference on Web Engineering*, Springer, 2010, pp. 85–90 (cit. on p. 10).

[30] M. A. Ghazanfar and A. Prugel-Bennett, ‘A scalable, accurate hybrid recommender system,’ in *2010 Third International Conference on Knowledge Discovery and Data Mining*, IEEE, 2010, pp. 94–98 (cit. on pp. 10, 11).

[31] C.-N. Ziegler, G. Lausen and L. Schmidt-Thieme, ‘Taxonomy-driven computation of product recommendations,’ in *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, 2004, pp. 406–415 (cit. on p. 10).

[32] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller and J. Riedl, ‘Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system,’ in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, 1998, pp. 345–354 (cit. on pp. 11, 14).

[33] R. Burke, *The adaptive web: Methods and strategies of web personalization. lncs, vol. 4321*, 2007 (cit. on pp. 11, 14).

[34] P. Cunningham, R. Bergmann, S. Schmitt, R. Traphöner, S. Breen and B. Smyth, ‘Websell: Intelligent sales assistants for the world wide web,’ *KI*, vol. 15, no. 1, pp. 28–32, 2001 (cit. on pp. 11, 14).

[35] I. Konstas, V. Stathopoulos and J. M. Jose, ‘On social networks and collaborative recommendation,’ in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 195–202 (cit. on pp. 11, 14).

[36] D. H. Lee and P. Brusilovsky, ‘Social networks and interest similarity: The case of citeulike,’ in *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, 2010, pp. 151–156 (cit. on pp. 11, 14).

[37] M. K. Condliff, D. D. Lewis, D. Madigan and C. Posse, ‘Bayesian mixed-effects models for recommender systems,’ in *ACM SIGIR*, Citeseer, vol. 99, 1999, pp. 23–30 (cit. on p. 11).

[38] J. Delgado and N. Ishii, ‘Memory-based weighted majority prediction,’ in *SIGIR Workshop Recomm. Syst. Citeseer*, Citeseer, 1999, p. 85 (cit. on p. 14).

[39] T. Vasista and M. A. AlSudairi, ‘Service-oriented architecture (soa) and semantic web services for web portal integration,’ in *Advances in Computing and Information Technology*, Springer, 2013, pp. 253–261 (cit. on p. 15).

[40] K. N. Jain, V. Kumar, P. Kumar and T. Choudhury, ‘Movie recommendation system: Hybrid information filtering system,’ in *Intelligent Computing and Information and Communication*, Springer, 2018, pp. 677–686 (cit. on p. 16).

[41] M. Braunhofer, V. Codina and F. Ricci, ‘Switching hybrid for cold-starting context-aware recommender systems,’ in *Proceedings of the 8th ACM Conference on Recommender systems*, 2014, pp. 349–352 (cit. on p. 16).

[42] A. S. Lampropoulos, P. S. Lampropoulou and G. A. Tsihrintzis, ‘A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis,’ *Multimedia Tools and Applications*, vol. 59, no. 1, pp. 241–258, 2012 (cit. on p. 16).

[43] E. Çano and M. Morisio, ‘Hybrid recommender systems: A systematic literature review,’ *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1487–1524, 2017 (cit. on p. 17).

[44] M. Ayub, M. A. Ghazanfar, M. Maqsood and A. Saleem, ‘A jaccard base similarity measure to improve performance of cf based recommender systems,’ in *2018 International Conference on Information Networking (ICOIN)*, IEEE, 2018, pp. 1–6 (cit. on p. 18).

[45] T. Q. Lee, Y. Park and Y.-T. Park, ‘A similarity measure for collaborative filtering with implicit feedback,’ in *International Conference on Intelligent Computing*, Springer, 2007, pp. 385–397 (cit. on p. 18).

[46] S. Bansal, C. Gupta and A. Arora, ‘User tweets based genre prediction and movie recommendation using lsi and svd,’ in *2016 Ninth International Conference on Contemporary Computing (IC3)*, IEEE, 2016, pp. 1–6 (cit. on p. 18).

[47] S.-M. Choi, S.-K. Ko and Y.-S. Han, ‘A movie recommendation algorithm based on genre correlations,’ *Expert Systems with Applications*, vol. 39, no. 9, pp. 8079–8085, 2012 (cit. on p. 19).

[48] S. Bag, S. K. Kumar and M. K. Tiwari, 'An efficient recommendation generation using relevant jaccard similarity,' *Information Sciences*, vol. 483, pp. 53–64, 2019 (cit. on p. 19).

[49] D. Goldberg, D. Nichols, B. M. Oki and D. Terry, 'Using collaborative filtering to weave an information tapestry,' *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992 (cit. on p. 21).

[50] B. Pang, L. Lee and S. Vaithyanathan, 'Thumbs up? sentiment classification using machine learning techniques,' *arXiv preprint cs/0205070*, 2002 (cit. on p. 21).

[51] P. D. Turney, 'Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews,' *arXiv preprint cs/0212032*, 2002 (cit. on p. 21).

[52] M. Hu and B. Liu, 'Mining and summarizing customer reviews,' in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2004, pp. 168–177 (cit. on p. 21).

[53] B. Pang and L. Lee, 'Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales,' *arXiv preprint cs/0506075*, 2005 (cit. on p. 21).

[54] L. S. Gallege and R. R. Raje, 'Towards selecting and recommending online software services by evaluating external attributes,' in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, 2016, pp. 1–4 (cit. on p. 23).