Personalized Movie Recommendation with Item-Based Collaborative Filtering

By

SIT YIE SIAN

FACULTY OF COMPUTING AND INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY OF MANAGEMENT AND TECHNOLOGY KUALA LUMPUR

ACADEMIC YEAR
2022/2023

PERSONALIZED MOVIE RECOMMENDATION WITH ITEM-BASED COLLABORATIVE FILTERING

By

SIT YIE SIAN

Supervisor: Mr. Choong Yun Loong

A project report submitted to the

Faculty of Computing and Information Technology

in partial fulfillment of the requirement for the

Bachelor of Computer Science (Honours) in Data Science

Tunku Abdul Rahman University of Management and Technology

Department of Mathematical And Data Science

Faculty of Computing and Information Technology

Tunku Abdul Rahman University of Management and Technology

Kuala Lumpur

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University rules and regulations and would be subjected to disciplinary actions.

____yiesian_______

SIT YIE SIAN

ID: 21WMR03693

DATE: 28/4/2023

Abstract

The primary objective of this project is to design and implement a movie recommendation system using item-based collaborative filtering, enhancing users' movie-watching experience by providing personalized movie recommendations. Addressing the challenge of finding relevant and enjoyable movies amidst the rapid growth of the entertainment industry, our system caters to individual preferences and tastes. The project encompasses the development of an item-based collaborative filtering algorithm, collection and preprocessing of movie rating data from the MovieLens dataset, and integration of additional movie metadata from IMDb and Wikiwand through web scraping techniques. Employing tools and techniques such as Python programming, pandas, NumPy, and BeautifulSoup4 libraries, and similarity metrics like Pearson correlation coefficient and cosine similarity, the system incorporates various functional modules, including data preprocessing, similarity computation, rating prediction, and recommendation generation. Performance evaluation using metrics like Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), precision, recall, F1 Score, and Normalized Discounted Cumulative Gain (NDCG) demonstrates the effectiveness of the item-based collaborative filtering algorithm in generating personalized movie recommendations, successfully capturing individual user preferences and improving the movie-watching experience. Further improvement could address data sparsity issues and incorporate hybrid filtering techniques to enhance recommendation quality.

Keywords: item-based collaborative filtering, movie recommendation system, similarity computation, rating prediction

Acknowledgement

I would like to express my sincere gratitude to all those who have contributed to the successful completion of this project. Their valuable insights, ideas, and support have been instrumental in shaping the development and implementation of the movie recommendation system.

First and foremost, I would like to thank my classmate, whose invaluable ideas and suggestions have significantly contributed to the project's direction and design. Their enthusiasm and dedication to the project have not only helped me overcome various challenges but also inspired me to think critically and creatively throughout the process.

I also extend my appreciation to my supervisor and moderators, who have provided me with the necessary knowledge and guidance to navigate the complexities of building a recommendation system. Their expertise and experience have been crucial in shaping my understanding of the subject matter and developing a robust and effective system.

Additionally, I would like to acknowledge the contributions of the GroupLens Research team for providing the MovieLens dataset, which served as the foundation for our recommendation system. Their efforts in creating and maintaining this comprehensive dataset have greatly facilitated our work and enabled us to focus on the development and optimization of the recommendation algorithm.

Finally, I would like to thank my friends and family for their unwavering support and encouragement throughout the project. Their belief in my abilities and their constant motivation have been essential in helping me persevere and achieve my goals.

To all of you, thank you for your invaluable contributions to this project. Your collective efforts have made it possible for us to create a movie recommendation system that can enhance the movie-watching experience for countless users.

Table of Contents

Chapter 1

Introduction

1.0 Introduction

The frequency we use with media has increased significantly in the digital age, and movies are no exception. Making a choice from the vast selection of movies accessible on numerous platforms might be difficult. This is where movie recommender systems come in to solve the issue. Movie Recommender systems help users discover new movies that they are likely to enjoy based on their previous viewing history, ratings, and preferences.

For movie fans, movie recommendation systems are essential. By making it simple for people to discover new movies that are pertinent to their interests, it saves users' time and effort. With recommender systems, users can find movies that they might not have found on their own. As a result, users are more likely to consume material that is pertinent to their interests, increasing engagement and happiness with the media service.

Movie recommender systems are important because they provide personalized recommendations. As each user has unique likes and preferences, a one-size-fits-all strategy is ineffective. It can offer specialized recommendations by utilizing our recommender system to match the user's particular interests. Users enjoy the service more, are more engaged, and have a better overall experience as a result.

1.1 Project Objectives

1.1.1 Provide customize recommendation that match user interests

By using collaborative filtering, the recommender system can analyze and compare the ratings that users have given to suggest different movies that align with their user's preferences.The recommender algorithm looks for films that the user will enjoy in order to improve their overall movie-finding and viewing experience. In order to accomplish this goal, a dataset of movie ratings will be collected, and this dataset will be used to train a model that can forecast the ratings of related movies. We then will use the model to make recommendations to users that have received high predicted ratings from the model.

Increased user engagement and retention result from personalized recommendations. If suggestions are made based on the interests of the users, they are more likely to use the site for longer, see more content, and have a better overall experience. This leads to an increase in loyalty and retention because users are more inclined to stick with the service if they are satisfied with the recommendations offered.

1.1.2 Generate revenue for the content providers

Not only is it advantageous to users to receive personalized recommendations, but it can also bring in money for content providers.By using the personalized recommendations, content providers can increase user interaction with their platform. Users are therefore more inclined to continue with the service and may even spend more money to purchase additional material, which can increase providers' revenue. The method can also be utilized to present particular movies to consumers who would not have otherwise known about them. This might result in more people watching and renting these movies, which would enhance the providers' income. In addition, content providers can use the data to learn more about consumer patterns and preferences. Their choice of content and marketing tactics can be improved by using information, resulting in better income. Ultimately, by keeping consumers engaged with their service for longer periods of time, recommending movies to users, the movie recommendation engine that we are developing has the potential to bring in money for content providers.

1.1.3 Improving the value proposition of the media service

Making personalized recommendations improves the media service's value proposition. The service offers a more individualized and carefully curated experience by customizing recommendations to the user's tastes. This fosters a sense of worth and comprehension in the user, which can boost brand loyalty and encourage effective word-of-mouth advertising. In order to enhance views, fresh content or content that may not have received as much attention can be utilized to be promoted. By providing a personalized and intuitive recommendation system, the media service can differentiate itself by attracting new users from competitors for a more tailored and engaging movie-watching experience.

To enhance the value proposition of the media service is to provide users with additional content related to the movies that they are interested in. A watchlist function that enables users to save movies they are interested in watching might be built into the system. Based on the user's saved movies, the system can then use this watchlist data to offer even more tailored recommendations.

1.2 Problem Statement

1.2.1 Problem Statement

With so much information available in the modern world, it can be challenging to select movies that a person enjoys. Recommender systems are perhaps the most often used tool in data mining techniques. By developing a recommendation system that proposes movies based on what other people with similar tastes like, this project seeks to make it easy to locate good movies. Using the least amount of information feasible on a product's features, recommender systems aim to make locating a good or service easier. A number of factors are used to look at the relationships between patterns and user attributes in order to choose the best movie suggestions for a user.

1.2.2 Proposed Solution

The project's goal is to create a movie recommendation system that uses item-based collaborative filtering to provide users with personalized suggestions. This system's objective is to help users in selecting movies that suit their tastes in the current digital environment. By looking at the ratings that users have given to various movies and figuring out the similarity between pairs of movies based on these ratings, the system can offer recommendations to users based on their prior ratings and the ratings of similar movies. To do this, a dataset of movie reviews will be collected and used to train a machine learning model that can predict the ratings that consumers would give to films based on the ratings of comparable films. After the model has been trained, users will receive individualized recommendations based on their watching preferences and interests.

Collaborative filtering works by matching the similarities in items and users. It examines the user attributes along with the features of the previous content that users have viewed or looked up (Shen J., 2020). Recommendations in movie recommendation systems are based on user data and what other users with similar user data are watching. For instance, user demographics like age, gender, and ethnicity are selected by collaborative filtering in movie recommender systems (Dakhel G.M., 2011). With the use of these attributes, movie recommendations are created that match to individuals with similar demographic traits and past user search history. If the user doesn't input any data or there isn't enough data for any reliable clustering, collaborative filtering suffers from a cold start.

1.3 Background

In today's world, it can be hard to find content, like books, videos, and movies, that you like. To help people find what they want, some companies use a system called a recommender system. This system suggests content to users based on what they have liked in the past. There are two main ways to build a recommender system: collaborative filtering and content-based filtering. I propose a personalized movie recommendation system that utilizes item-based collaborative filtering. A strategy used frequently in recommendation systems is collaborative filtering, which includes examining the ratings that users have given to products and using this data to suggest products to other users according to their common rating patterns. A variation of collaborative filtering called item-based collaborative filtering emphasizes interconnections between various objects rather than connections between users and items.

1.4 Advantages and Contribution

There are several advantages that a movie recommendation system that utilizes item-based collaborative filtering could offer over competing systems. One of the key advantages of collaborative filtering approaches, including item-based collaborative filtering, is their ability to provide personalized recommendations to users based on their past ratings and the ratings of similar users or items. This allows the system to make more relevant and accurate recommendations to users, which can improve the overall user experience. They can make a real quality assessment of items by considering other people's experience (Sharma and Yadav).

Collaborative filtering approaches are generally very scalable, as they do not require the system to have detailed information about the attributes of the items being recommended (such as genre, actors, and plot keywords). This makes it easier to add new items to the system and maintain the recommendations over time. For systems that need detailed information, movie catalogs can rapidly change as time goes and this increases the difficulties as it requires the system to have a comprehensive and up-to-date database of item attributes.

1.5 Project Plan

1	Define the project scope and requirements
2	Collect and preprocess data
3	recommendation algorithm
4	Build and train the model
5	Implement the recommendation system
6	Test and validate the system
7	Deploy and maintain the system

1.5.1 Milestones

The proposed project milestone is as follows:

Milestone	Milestone Goal	Deadline
Concept approval	Feasibility studies and basic system concepts is approved	16/2/2023
Requirements review	Requirements specifications are complete, correct, approved and suitable for input to design.	26/2/2023
System design review	The system design satisfies all product requirements, is approved and is suitable for input into the detailed design process.	11/3/2023

UI design review	Detailed UI design of the system architecture, are approved and are suitable for input into the development of code.	18/3/2023
Test plan review	Test plans are adequate for the testing of all product features, are approved and are suitable for input to the development of test cases and test procedures.	22/5/2023
Test readiness review	Developed and unit tested software has been passed by the test team and is suitable for input into integration testing.	3/7/2023
System test review	The software product has passed system testing and is suitable for input into acceptance testing.	10/7/2023
Operational readiness	The software product has passed acceptance testing and is suitable for deployment in its target production environment.	24/7/2023

1.6 Project Team and Organization

System Modules	*Sit Yie Sian*	*Leong Sheng Mou*
Content-based Filtering		✓

Collaborative Filtering	✓	

1.7 Chapter Summary & Evaluation

In this chapter, as a summary, the main objectives of the system were discussed. Objectives that have been listed are to automate the coordination tasks in the movie recommended system, provide better and scalable movie rating reports from reviews . Besides, the problem statements of the current movie recommended system and the proposed solution were discussed. Furthermore, the background of the recommended system, the benefits of the movie recommended system were mentioned. Lastly, the project milestone schedule and project team and organization have been discussed.

Chapter 2

Literature Review

2 Literature Review

Popularity of the world is only increasing and systems for recommending movies have become a crucial component of the modern digital environment. We find it challenging to choose the movie we want to see because there are so many options available. For this reason, we require a recommender system to do it for us. Recommender systems have different methods to make movie recommendations to users including content-based filtering, collaborative filtering, and hybrid filtering. One of the best methods for creating movie recommendation systems is collaborative filtering, particularly item-based collaborative filtering. We will examine the idea of item-based collaborative filtering, how it is used in systems that recommend movies, as well as some of the benefits and drawbacks of this method in this literature review.

2.1 Item-Based Collaborative Filtering

An approach used in recommender systems to estimate user preferences for items based on their similarities to other items is called item-based collaborative filtering. Based on the user's rating history, the system creates an item similarity matrix. After that, each user will receive personalized recommendations based on the similarity matrix. The fundamental assumption behind item-based collaborative filtering is that users who have rated similar movies in the past will also rate similar movies a similar score. In other words, the system can predict a user's rating for a movie from the history of the user who has rated the similar movie.

The fact that item-based collaborative filtering can manage the sparsity of the user-item ratings matrix is one of its main features. For instance, a research by Zhou et al. (2010) has compared the effectiveness of their approach to a number of different recommendation techniques, such as matrix factorization and user-based collaborative filtering. The result demonstrated that their research revealed that item-based collaborative filtering outperformed the other ones in terms of managing the sparsity of the user-item rating matrix. In particular, whereas the other approaches struggled with sparsity, item-based collaborative filtering was able to offer correct recommendations for individuals who had only reviewed a few movies. Overall, the authors' findings shows that Item-based collaborative filtering appears to be a potential strategy for movie selection that can assist in managing sparsity in user-item rating matrices.

Item-based collaborative filtering also has the benefit of being able to deal with cold-start issues. For instance, a research by Sarwar et al. (2001) has demonstrated that item-based collaborative filtering algorithms perform better than various approaches, especially when it comes to tackling the cold-start problem. In particular, the algorithms can use the similarity between products to create precise suggestions for new users and new items in the absence of user data. Overall, the authors' findings are in line with the notion that cold-start problems in recommendation systems can be successfully addressed by item-based collaborative filtering.

It has also been demonstrated that item-based collaborative filtering is very scalable. For instance, a research by Lemire & Maclachlan (2005) has demonstrated that Slope One predictors can handle very large datasets with millions of users and items and are noticeably faster than other methods. The authors also show that their algorithm can make recommendations with little latency in real-time. Overall, according to the authors', the Item-based collaborative filtering, particularly in the form of Slope One predictors, is extremely scalable and can successfully handle massive datasets with millions of users and objects. They evaluated how well their algorithm performs in comparison to other recommendation techniques, such as traditional item-based collaborative filtering.

2.2 Item-Based Collaborative Filtering in Movie Recommendation Systems

Several researches have demonstrated that in movie recommendation systems, item-based collaborative filtering performs better than other strategies like content-based filtering. For instance, a research by Sarwar et al. (2001) demonstrated that in terms of prediction accuracy, item-based collaborative filtering beats content-based filtering. According to the study, item-based collaborative filtering was more accurate than content-based filtering at predicting user ratings.

In a further study, Shao et al. (2019) evaluated the effectiveness of deep learning, matrix factorization, and item-based collaborative filtering. According to the study, even when the dataset was limited, item-based collaborative filtering could still produce reliable recommendations. Also, in terms of prediction accuracy, item-based collaborative filtering was able to outperform matrix factorization and deep learning techniques.

Using item-based collaborative filtering in movie recommendation systems can be done in a number of ways. To measure how similar two movies are, a typical method is to apply the Pearson correlation coefficient or cosine similarity. After that, recommendations are generated for each user based on the similarity matrix.

Another strategy is to group similar movies together using clustering algorithms. The system can then produce suggestions based on the user's past viewing habits and movies in similar clusters to those they have watched before.

A third strategy is to divide the user-item ratings matrix into low-rank matrices using matrix factorization techniques like singular value decomposition (SVD) or non-negative matrix factorization (NMF). The low-rank matrices can then be used by the system to produce suggestions.

2.2.1 Limitations of Item-Based Collaborative Filtering

Item-based collaborative filtering has the drawback of being unable to deal with user bias. User bias is the propensity of some users to rate movies more favorably or unfavorably than other users. Item-based collaborative filtering makes the assumption that people who have previously rated movies similarly will continue to do so for new released movies. This assumption might not be accurate for people who have strong biases. For instance, a research by Shen et al. (2009) demonstrates that PMF-UB performs noticeably better than alternative approaches, especially when it comes to minimizing the effects of user bias. The authors show that the algorithm outperforms traditional item-based collaborative filtering in terms of accurately recommending movies to viewers even with strong biases. Overall, the authors' findings are consistent with the idea that item-based collaborative filtering may be a significant drawback due to user bias.

The inability of item-based collaborative filtering to capture the temporal dynamics of user preferences is another drawback. For instance, a research by Lamere (2013) has shown that user preferences are not constant and can vary over time, which could cause recommendations to become less accurate as time passes. The author proposes a new algorithm called Time-aware Item-based Collaborative Filtering (TiCF). The algorithm uses a weighted similarity score that gives greater weight to more recent ratings than to older ratings. The algorithm uses a decay function in order to gradually reduce the weight of older ratings. According to the experiments, TiCF performs noticeably better than conventional item-based collaborative filtering, especially as the dataset grows larger and more sparse.

The "long tail" issue also affects item-based collaborative filtering. The majority of the movies in the dataset have very few ratings, which is known as the "long tail problem" and makes it challenging to produce reliable recommendations for these movies. For instance, a research by Burke (2007) analyzes a number of sizable datasets, including the Netflix Prize dataset, which comprises user ratings for more than 100 million movies. The research reveals that most of the movies in the dataset have a very small number of ratings, which makes it difficult to generate reliable recommendations for these movies.

2.3 Chapter Summary and Evaluation

In conclusion, Item-based collaborative filtering has become one of the best approaches for movie recommendation systems. It is very scalable, can manage cold-start issues, and can handle the sparsity of the user-item ratings matrix. In terms of prediction accuracy, it has been demonstrated that item-based collaborative filtering performs better than other methods like content-based filtering.

Item-based collaborative filtering has certain drawbacks, such as the inability to handle user bias, capture the temporal dynamics of user preferences, and address the long tail. However, it is still a well-liked method for movie recommendation systems due to its efficiency and scalability, despite these drawbacks.

Chapter 3

Methodology and Requirements Analysis

3 Methodology and Requirements Analysis

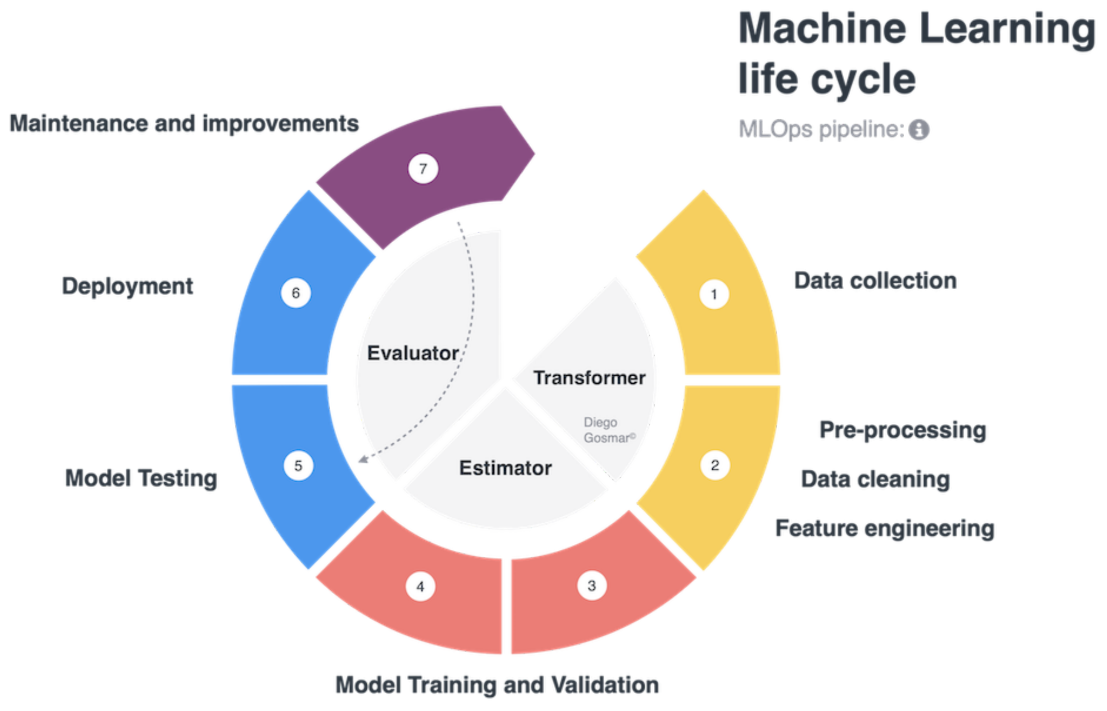


Figure 3.1 Machine Learning Pipeline

3.1 Introduction

This chapter will outline the methodology and requirements analysis for a general machine learning pipeline, focusing on the key stages involved in the design, implementation, and evaluation of a system. The chapter will discuss data collection, data preprocessing, model selection, training, evaluation, and deployment of the system. Additionally, the chapter will outline evaluation metrics and methods for measuring the effectiveness of the model.

3.2 Data Collection

Data collection is the first and one of the most critical stages in the machine learning pipeline. The data collected should be representative of the problem domain and provide sufficient information to train and evaluate the model. Data collection can involve multiple sources, such as public datasets, proprietary databases, user-generated content, or data obtained through web scraping. When collecting data, it is crucial to consider factors like data quality, size, diversity, and relevance to the problem at hand.

3.3 Data Preprocessing

Data preprocessing is an essential step in preparing the data for analysis and model training. The main goal of data preprocessing is to clean, transform, and structure the data, ensuring that it is consistent and suitable for further processing. Typical data preprocessing steps include:

- Data integration: Combining data from multiple sources into a unified dataset.
- Handling missing values: Identifying and addressing missing data points using techniques like imputation or removal.
- Data transformation: Converting data into a suitable format for analysis, such as normalizing numerical data or encoding categorical features.
- Feature engineering: Creating new features or modifying existing ones to improve model performance and interpretability.
- Data partitioning: Splitting the dataset into training, validation, and testing sets to evaluate model performance and prevent overfitting.

3.4 Model Selection and Training

Model selection involves choosing the most appropriate machine learning algorithm for the problem at hand. This process typically requires a thorough understanding of the problem domain, the data, and the available algorithms. Factors to consider when selecting a model include:

- Algorithm type: Choosing between supervised, unsupervised, or reinforcement learning methods, depending on the problem's requirements.
- Model complexity: Balancing between model simplicity and flexibility to prevent underfitting or overfitting.
- Scalability: Ensuring the selected algorithm can handle the dataset size and complexity.
- Interpretability: Prioritizing models that provide clear insights into the relationships between features and the target variable.

Once the model has been selected, the training process involves using the training dataset to adjust the model's parameters so that it can generalize well to unseen data. This may involve techniques like gradient descent, regularization, or other optimization methods.

3.5 Model Evaluation

Evaluating the performance of the machine learning model is crucial to ensure that it meets the desired objectives and provides accurate and reliable predictions. Evaluation metrics and methods help quantify the model's performance and facilitate comparisons with alternative approaches. Common evaluation metrics include:

- Accuracy: The proportion of correct predictions made by the model.
- Precision, Recall, and F1 Score: Metrics that consider the balance between true positives, false positives, and false negatives.
- Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE): Metrics that measure the difference between predicted and actual values for regression tasks.
- Area Under the Receiver Operating Characteristic (ROC) Curve (AUC-ROC): A metric that evaluates the trade-off between true positive rate and false positive rate.

Cross-validation is a common technique used to evaluate model performance by partitioning the dataset into multiple folds and iteratively training and testing the model on different subsets.

3.6 Deployment and Maintenance

Once the model has been trained and evaluated, it is ready for deployment in a production environment. The deployment process involves integrating the model into a larger system,such as a web application, mobile app, or other software platforms. Deployment considerations include:

- Integration: Ensuring that the model can be seamlessly integrated into the target system and interact with other components.

- Scalability: The deployed model should be able to handle varying loads and adapt to changes in the data and user requirements.
- Latency: Ensuring the model can provide timely predictions or recommendations, considering the real-time requirements of the application.
- Security: Protecting the model and its data from unauthorized access or tampering.

Maintenance is an ongoing process that involves monitoring the performance of the deployed model, updating it with new data, and fine-tuning the model to address any changes in the problem domain or user requirements. Regular maintenance helps ensure that the machine learning system remains accurate, reliable, and relevant over time.

3.7 Ethics Consideration

Ethics play a significant role in the development and deployment of machine learning systems, as these systems have the potential to impact users and society in various ways. It is essential to consider ethical implications throughout the machine learning pipeline to ensure that the developed models are fair, transparent, and respectful of user privacy. In this section, we will discuss some key ethical considerations for machine learning systems.

3.7.1 Data Privacy and Security:

When collecting and processing data, it is crucial to protect user privacy by handling sensitive information responsibly. This includes anonymizing data, obtaining user consent, and complying with relevant data protection regulations, such as the General Data Protection Regulation (GDPR). Additionally, data storage and access should be secured to prevent unauthorized access and data breaches.

3.7.2 Bias and Fairness:

Machine learning models can inadvertently learn and perpetuate biases present in the training data, leading to unfair treatment of certain user groups. To address this issue, researchers and practitioners should actively identify and mitigate potential biases in the data and model. Techniques such as re-sampling, re-weighting, and adversarial training can help reduce bias and promote fairness in the model's predictions.

3.7.3 Transparency and Explainability:

Machine learning models, particularly deep learning models, can often be seen as "black boxes" that produce predictions without a clear explanation of how they arrived at those results. Ensuring transparency and explainability in machine learning systems can help users understand the decision-making process, build trust in the system, and facilitate accountability. Techniques such as interpretable models, feature importance analysis, and model-agnostic explanation methods can help improve transparency and explainability in machine learning systems.

By addressing these ethical considerations throughout the machine learning pipeline, developers can create systems that are more responsible, fair, and user-centric. Ensuring that machine learning systems adhere to ethical principles not only benefits the end-users but also helps build trust in the technology and fosters long-term adoption and success.

3.8 Chapter Summary

In this chapter, we have provided an overview of the methodology and requirements analysis for a general machine learning pipeline. The stages involved in the pipeline include data collection, data preprocessing, model selection and training, evaluation, and deployment and maintenance. Each of these stages plays a crucial role in the development and implementation of an effective machine learning system.

By following the steps and techniques discussed in this chapter, researchers and practitioners can create machine learning systems that cater to the specific needs of their application domain, ensuring that the developed solutions are accurate, reliable, and efficient. This general framework serves as a foundation for building various types of machine learning systems, ranging from recommendation engines and natural language processing models to computer vision and reinforcement learning applications.

Chapter 4

System Design

4.0 System Design

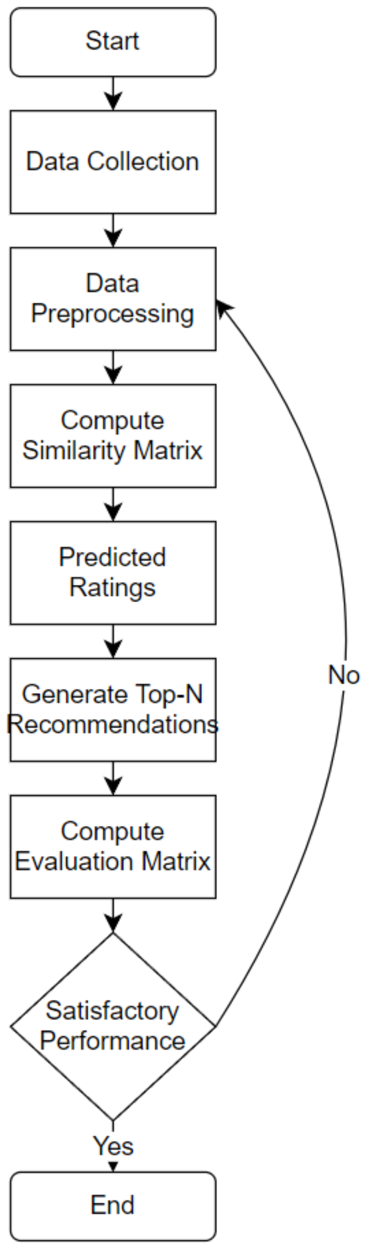


Figure 4.1 Item-Based Collaborative Filtering Pipeline

4.1 Introduction

This chapter will outline the system design for the proposed movie recommendation system that utilizes item-based collaborative filtering. The chapter will discuss the data collection process, data preprocessing, and the implementation of the item-based collaborative filtering algorithm. Additionally, the chapter will outline evaluation metrics and methods for measuring the effectiveness of the recommendation system.

4.2 Data Collection

To train and evaluate the item-based collaborative filtering algorithm, a dataset containing user-movie ratings is required. The MovieLens dataset provided by GroupLens Research is an appropriate choice for this project. The dataset consists of user ratings for various movies, along with movie metadata such as genres, titles, and release years. The dataset is publicly available, and it contains multiple versions with different sizes, allowing for experiments with varying levels of sparsity and scalability.

In addition to the MovieLens dataset, we also use the BeautifulSoup4 library to scrape additional movie data from IMDb and Wikiwand to enhance our item-based collaborative filtering system. This section will introduce BeautifulSoup4, IMDb, and Wikiwand, and explain why we use the MovieLens dataset.

BeautifulSoup4 is a popular Python library for web scraping, which allows for extracting data from HTML and XML documents. It is designed to parse web pages and make it easy to navigate, search, and modify the parse tree. BeautifulSoup4 provides a convenient way to access the tags within an HTML or XML document and extract relevant information for further analysis. By using BeautifulSoup4, we can efficiently scrape data from IMDb and Wikiwand to complement the MovieLens dataset.

IMDb (Internet Movie Database) is an online database that contains a wealth of information about movies, TV shows, and other forms of visual media. IMDb offers details about cast and crew, production information, ratings, reviews, and other related data. We use IMDb in our data collection process to obtain additional movie metadata, such as director, actors, release year, and genres, which can be useful for refining our item-based collaborative filtering model.

Wikiwand is a modern interface for browsing Wikipedia content. It enhances the readability and presentation of Wikipedia articles and offers an improved user experience. We use Wikiwand to scrape additional movie information, such as plot summaries and other contextual data, to augment our movie recommendation system. Incorporating this information into our item-based collaborative filtering model can help improve the quality of recommendations by providing a more comprehensive understanding of the movies.

We use the MovieLens dataset as our primary data source because it is a well-structured and widely used dataset for building movie recommendation systems. The MovieLens dataset contains user ratings for a large number of movies, which are essential for collaborative filtering methods. It also includes movie metadata, such as genres and title, which can be used for content-based filtering or hybrid methods. By combining the data from MovieLens, IMDb, and Wikiwand, we can create a more robust and comprehensive dataset for building our item-based collaborative filtering system.

4.3 Data Preprocessing

Data preprocessing is a crucial step in building a recommendation system, as it ensures that the collected data is clean, consistent, and suitable for analysis. In the context of our item-based collaborative filtering system, the data preprocessing stage involves the following steps:

1. Data integration: Since we gather data from multiple sources, such as the MovieLens dataset, IMDb, and Wikiwand, it is essential to combine this information into a unified dataset. This process involves merging and aligning the data based on common identifiers, such as movie titles or movie IDs, and resolving any discrepancies between the sources.

2. Handling missing values: Missing data can lead to inaccurate recommendations or introduce biases in the model. We need to identify and handle missing values in the dataset. Common techniques for dealing with missing data include:

a. Removing records with missing values.

b. Filling missing values with the mean or median value of the respective feature.

c. Using interpolation or regression techniques to estimate missing values based on other available data.

3. Data transformation: Transforming the data into a suitable format for analysis is essential. This can include:

a. Converting textual data, such as genres or plot summaries, into numerical representations using techniques like one-hot encoding or natural language processing methods (e.g., TF-IDF or word embeddings).

b. Normalizing numerical data, such as user ratings or release years, to ensure that all features have a similar scale and do not introduce biases in the model (e.g., Min-max scaling, Z-score normalization or Log transformation).
c. Aggregating data, if necessary, to create new features or reduce the dimensionality of the dataset. (e.g., PCA or SVD)

4. Feature engineering: Creating new features or modifying existing ones can help improve the performance of the recommendation system. Some examples of feature engineering include:
   a. Extracting relevant information from text data, such as keywords or named entities, which can be used as additional features for the similarity calculation.

b. Computing interaction features, such as the number of common users who rated a pair of movies or the average rating difference between the movies.

5. Data partitioning: Splitting the dataset into training and testing sets is necessary to evaluate the performance of the recommendation system. This ensures that the model is validated on unseen data and helps prevent overfitting. Typically, a ratio of 70-30 or 80-20 is used for partitioning the dataset, with the larger portion used for training and the smaller portion used for testing.

By performing these data preprocessing steps, we can ensure that our item-based collaborative filtering system is built on clean, consistent, and relevant data, which will help improve the accuracy and reliability of the movie recommendations.

4.4 Implementation of Item-Based Collaborative Filtering

The implementation of the item-based collaborative filtering algorithm involves the following steps:

1. Similarity Computation:

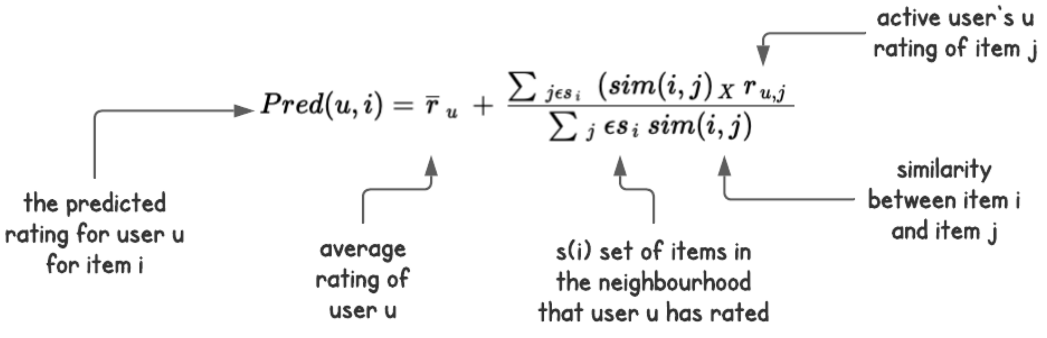
To calculate the similarity between pairs of movies, a similarity metric is used. Common similarity metrics include:

- Pearson correlation coefficient: This measures the linear correlation between two variables (in this case, movie ratings). The coefficient ranges from -1 (negative correlation) to 1 (positive correlation), with 0 indicating no correlation.
- Cosine similarity: This calculates the cosine of the angle between two non-zero vectors (movie rating vectors). The similarity ranges from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating no similarity.

The similarity computation results in a movie similarity matrix, where each cell represents the similarity between a pair of movies.

2. Rating Prediction:

After computing the similarity matrix, the next step is to predict the ratings for the movies that a user has not yet rated. For each user, the predicted rating for an unrated movie is calculated as a weighted sum of the user's ratings for similar movies, where the weights are the similarity values between the unrated movie and the rated movies.
The predicted rating (PR) for user 'u' and movie 'i' can be calculated using the following formula:


active user's u rating of item j
the predicted rating for user u for item i
average rating of user u
s(i) set of items in the neighbourhood that user u has rated
similarity between item i and item j

$$Pred(u,i) = \bar{r}_u + \frac{\sum_{j \epsilon s_i} (sim(i,j) \, x \, r_{u,j}}{\sum_{j} \epsilon s_i \, sim(i,j)}$$

where 'sim(i, j)' represents the similarity between movie 'i' and movie 'j', and R(u, j) is the user 'u's rating for movie 'j'. The summation is over all movies 'j' that user 'u' has rated.

3. Top-N Recommendations:

Once the predicted ratings have been calculated for all unrated movies, the algorithm generates a list of top-N recommendations for each user. This list contains the N movies with the highest predicted ratings for that user. The recommendations are personalized and based on the user's previous movie ratings and the similarities between movies.
In practice, the recommendations might be further refined by considering additional factors such as movie popularity, user preferences, or diversity in genres.
In summary, the item-based collaborative filtering algorithm calculates movie similarities, predicts user ratings for unrated movies, and generates personalized top-N recommendations for each user based on those predictions.

4.5 Evaluation Metrics and Methods

In the context of movie recommendation systems, it's essential to evaluate the performance of the implemented algorithm to ensure it's providing accurate and relevant recommendations to users. Evaluation metrics and methods help quantify the algorithm's performance and can be used for comparison with alternative approaches. Here are some commonly used evaluation metrics and methods for recommendation systems:

1. Mean Absolute Error (MAE):

Mean Absolute Error measures the average absolute difference between the predicted ratings and the actual user ratings. It's a straightforward metric to calculate and interpret. A lower MAE value indicates better prediction accuracy.

2. Root Mean Squared Error (RMSE):

Root Mean Squared Error is the square root of the average squared difference between the predicted ratings and the actual user ratings. It penalizes larger errors more heavily than smaller errors, making it more sensitive to outliers. A lower RMSE value indicates better prediction accuracy.

3. Precision and Recall:

Precision measures the proportion of relevant recommendations among all recommendations provided by the system, while recall measures the proportion of relevant recommendations retrieved out of all relevant items in the dataset. These metrics are commonly used together to evaluate the trade-off between providing a high number of relevant recommendations (recall) and minimizing the number of irrelevant recommendations (precision).

4. F1 Score:

The F1 Score is the harmonic mean of precision and recall, which provides a single metric to evaluate the balance between precision and recall. A higher F1 Score indicates better performance.

5. Normalized Discounted Cumulative Gain (NDCG):

NDCG is a ranking-based evaluation metric that measures the usefulness of recommended items, considering their position in the recommendation list. It takes into account the relevance of each recommended item and discounts the relevance score of items ranked lower in the list. A higher NDCG value indicates better recommendation quality.

To evaluate the performance of the proposed recommendation system, a k-fold cross-validation approach can be used. The dataset will be divided into k equally sized folds, and the algorithm will be trained on k-1 folds and tested on the remaining fold. This process will be repeated k times, with each fold serving as the test set once. The average performance across all k iterations will be reported.

4.6 Chapter Summary

In this chapter, we have discussed the methodology and requirements analysis for implementing an item-based collaborative filtering movie recommendation system. We began by highlighting the importance of data collection and the use of the MovieLens dataset, combined with web scraping techniques like BeautifulSoup4 to collect additional data from sources such as IMDb and Wikiwand. This combination of data sources provides a rich dataset to work with, capturing various aspects of movies that can be utilized in the recommendation algorithm.

The data preprocessing stage is crucial for cleaning and preparing the data for analysis. We discussed various preprocessing steps, such as handling missing values, normalizing numerical data using techniques like Z-score normalization or Min-Max scaling, and aggregating data to create a more comprehensive dataset. We also emphasized the importance of feature engineering, which involves extracting new features from the existing data and using natural language processing techniques to extract sentiment scores from movie reviews.

Feature selection is a critical step in building an effective recommendation system, as it helps to select the most relevant features for the algorithm while minimizing noise and overfitting. We mentioned the use of techniques like LASSO regression, which can help identify features with large coefficients, indicating their importance in the model.

Next, we elaborated on the implementation of item-based collaborative filtering, discussing the steps involved, such as computing movie similarity, predicting ratings, and generating top-N recommendations. We also explained the importance of evaluating the performance of the recommendation algorithm using various evaluation metrics, such as MAE, RMSE, precision, recall, F1 Score, and NDCG, and cross-validation.

In summary, this chapter provides a comprehensive overview of the methodology and requirements analysis for building an item-based collaborative filtering movie recommendation system. By following the steps and techniques discussed, researchers and practitioners can create an effective recommendation system that caters to the preferences and needs of individual users, ultimately improving their movie-watching experience.

References

1. Burke, R. (2007). The long tail problem in recommender systems. In Proceedings of the 1st ACM Conference on Recommender Systems (pp. 1-5). ACM.
2. Dakhel G.M., Mahdavi M. (2011). A new collaborative filtering algorithm using K-means clustering and neighbors' voting. In Proceedings of the 11th International Conference on Hybrid Intelligent Systems (HIS); Malacca, Malaysia. pp. 179–184.
3. Lamere, P. (2013). Modeling temporal dynamics of user preferences for improved recommendations. In Proceedings of the 7th ACM Conference on Recommender Systems (pp. 237-240). ACM.
4. Lemire, D., & Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering (2007). ResearchGate. Retrieved from https://www.researchgate.net/publication/1960789_Slope_One_Predictors_for_Online_Rating-Based_Collaborative_Filtering
5. Sarwar, B. et al. (2001). Item-based collaborative filtering recommendation algorithms: Proceedings of the 10th International Conference on World Wide Web, ACM Conferences. Available at: https://dl.acm.org/doi/10.1145/371920.372071
6. Shen, X., Liu, B., & Chen, H. (2009). Reducing the effects of user bias in collaborative filtering for personalization. In Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (pp. 784-791). Springer.
7. Shen J., Zhou T., Chen L. (2020). Collaborative filtering-based recommendation system for big data. International Journal of Computer Science and Engineering, 21, 219–225.
8. Shao, C., Wang, M., Lin, X., Wang, S., & Zhang, X. (2019). Comparative study of collaborative filtering, matrix factorization and deep learning techniques for recommender systems. Journal of Big Data, 6(1), 1-23.
9. Zhou, J., Wu, F., Cao, J., & Zhuang, Y. (2010). A novel approach for movie recommendation with item-based collaborative filtering. In Proceedings of the International Conference on Multimedia and Expo (ICME) (pp. 1764-1769). IEEE.

This page is intentionally left blank to indicate the back cover. Ensure that the back cover is black in color.