

A SURVEY OF DIFFERENT RECOMMENDATION ALGORITHMS AND THEIR APPLICATIONS

Thesis Submitted for the Partial Fulfilment of the Award of the

Degree of

Master of Technology

in

Industrial Mathematics and Scientific Computing

by

Deepanshu Tyagi

(MA21M009)

Under the Supervision of

Prof. S. Sundar

Dr. Sri Vallabha Deevi(Tiger Analytics)



Department of Mathematics
Indian Institute of Technology Madras,
Tamil Nadu, 600036, India

May 2023

INDIAN INSTITUTE OF TECHNOLOGY, MADRAS
DEPARTMENT OF MATHEMATICS

NAME : Deepanshu Tyagi

Roll No : MA21M009

PROGRAMME : M.Tech.

DATE OF JOINING : 28.07.2021

Guide : Prof. Dr. S. Sundar

Co-guide : Dr. Sri Vallabha Deevi

Certificate

This is to certify that the report “**A Survey of different recommendation algorithms and their applications**”, submitted by **Deepanshu Tyagi (MA21M009)**, in partial fulfillment of the requirement for the award of the degree of Master of Technology in Industrial Mathematics and Scientific Computing, Indian Institute of Technology Madras, is the record of the work done by him during the academic year 2022-2023 in the Department of Mathematics, IIT Madras, India, under my supervision.

Prof. Dr. S. Sundar
Department of Mathematics
IIT Madras

Dr. Sri Vallabha Deevi
Tiger Analytics
Chennai, India

Acknowledgement

I would like to thank Prof. Dr. S. Sundar and Dr. Sri Vallabha Deevi for their constant help and support, without which the completion of my project for this academic year would be impossible. They have always been there to guide me throughout the course and have patiently solved all the queries associated with the project. Lastly, I would like to thank the Department of Mathematics, IIT Madras and Tiger Analytics for providing me with the opportunity to work in this exciting field.

Deepanshu Tyagi MA21M009
Department of Mathematics
IIT Madras

Abstract

With the proliferation of digital communication and electronic commerce, it has become much easier for organizations to capture the most granular data about their customer's behavior. With this rich history, organizations can make relevant recommendations to their customers about new products, services, offers or discounts. Many recommendation algorithms are used for a wide variety of tasks.

This study performs a survey of recommendation algorithms. It attempts to classify recommendation tasks into different classes, based on the number of products, number of customers, one-time or repeat purchases, etc. An assessment of each class of recommendation algorithm on a relevant dataset is performed and the analysis is presented here.

Contents

1	Introduction	8
1.1	For industries	8
1.2	For users	9
2	Recommendation System	10
2.1	Information Filtering system	10
2.2	Recommendation System	10
2.3	Terminology	13
3	Types of Recommendation Systems	15
3.1	Non - Personalized Recommendation System	16
3.2	Personalized Recommendation System	16
3.2.1	Content-based Recommendation System	16
3.2.2	Collaborative filtering (CF) Recommendation System	17
3.2.3	Hybrid Recommendation System	20
4	Classifying Recommendation Systems	23
5	Evaluation metrics	26
5.1	Root Mean Squared Error (RMSE)	26
5.2	Precision at K	26
5.3	Recall at K	27
5.4	Area under the curve (AUC)	27
5.5	Accuracy	28

6	Models	29
6.1	Hybrid Model	29
6.1.1	Matrix factorization	30
6.1.2	LightFM	31
6.2	Generalized Matrix Factorization (GMF)	32
6.3	Neural Collaborative Filtering (NCF)	34
6.4	Neural Matrix Factorization(NeuMF)	35
6.5	Alternating Least Squares (ALS)	36
6.6	Deep Walk	37
7	Datasets and Implementation	39
7.1	Movie Lens Dataset	39
7.2	H & M Fashion Dataset	44
7.3	lastfm dataset	50
7.4	Dunnhumby dataset	54
8	Friend recommendations in Social Network	57
8.1	Facebook Friend Dataset	58
8.2	Using Graph Properties and Classification algorithms	59
8.3	Graph Neural Network (GNN)	63
8.3.1	Graph Convolutional Neural Network (GCN)	65
8.3.2	GraphSAGE (Graph SAmple and aggreGatE)	65
8.3.3	Results	66
8.4	Results	67
8.5	Conclusion :	67
9	Conclusions	68

Chapter 1

Introduction

The last few decades have witnessed an unprecedented rise in the utilization of online services such as YouTube, Amazon, Facebook, and Netflix. As a result, recommender systems have gained an increasingly significant role in our daily lives. Netflix employs these systems to recommend movies or web series to users based on their preferences. E-commerce platforms suggest articles that may interest potential buyers, while Facebook and other social networks recommend friends and people to follow. A recommendation system is an algorithm designed to suggest relevant items to users, such as movies to watch, texts to read, or clothes to buy.

1.1 For industries

The competition among businesses is challenging in today's digital world, and it is hard to attract customers. With so many options, businesses must provide personalized experiences to meet their customers' needs. That is where a recommendation system can help. A recommendation system uses advanced technology to analyze customer data and suggest personalized products or services to customers. This improves the customer experience, sets businesses apart from competitors, and helps increase revenue and customer loyalty.

Businesses can use a recommendation system to increase revenue by offering customers personalized products or services. By analyzing customer data, businesses can understand their customer's preferences and increase their purchase chances. The customer

data can also be used to create targeted marketing campaigns and promotions, which can be more effective.

A well-designed recommendation system can help businesses build a positive brand image by showing that they prioritize customer satisfaction and convenience. By providing personalized experiences, businesses can stand out from their competitors and build long-term customer relationships.

In summary, a recommendation system can be a game-changer for businesses in today's digital world. By providing personalized customer experiences, businesses can increase revenue, improve their marketing efforts, and enhance their brand image, leading to greater customer loyalty and satisfaction.

1.2 For users

A recommendation system can be beneficial when browsing websites. The system uses advanced technology to suggest high-quality items that match the user's preferences, making their online experience more enjoyable. One of the main benefits of a recommendation system is that it saves users time by reducing the need to search through many options. The system provides a curated list of items most likely to interest the user, making it easier to find what they need quickly.

Additionally, a recommendation system can help users discover new and exciting items they may not have found otherwise. For example, a movie recommendation system may suggest lesser-known films that align with the user's taste, expanding their viewing options beyond popular ones. Similarly, a product recommendation system may introduce users to unique and innovative products they may have missed during their search. Finally, a recommendation system can improve the quality of the items users are exposed to by filtering out irrelevant and low-quality items. The system presents users with a carefully curated selection of high-quality options.

In conclusion, a recommendation system is beneficial for users in many ways. By providing personalized and efficient experiences, saving time, introducing new items, and improving the quality of options, users can navigate the vast online world with greater ease and satisfaction.

Chapter 2

Recommendation System

2.1 Information Filtering system

An information filtering system is designed to manage the overwhelming amount of information available to users. Using (semi)automated or computerized methods, it removes redundant or unwanted information from an information stream before presenting it to the user. The system compares the user's profile to reference characteristics derived from the item information (content-based approach) or the user's social environment (collaborative filtering approach) to achieve this.

Machine learning algorithms, such as spam filtering, can be employed as information filtering algorithms. Active information filterings systems, such as recommender systems and content discovery platforms, seek to provide users with relevant information items, such as films, television, music, books, news and web pages, that match their interests and preferences. Unlike traditional filtering systems, these systems add information items to the user's information flow rather than removing them. Collaborative filtering approaches or a combination of collaborative and content-based filtering approaches are typically used in recommender systems, although content-based recommender systems exist.

2.2 Recommendation System

A recommendation system is an information filtering system that uses machine learning algorithms to predict and rank items or users based on their preferences. The system

analyzes user behavior, purchase history, ratings and other data to provide personalized suggestions for items a user might be interested in. The main goal of a recommendation system is to improve user engagement and satisfaction by providing relevant and personalized content.

Recommendation systems can be divided into two categories: collaborative filtering and content-based filtering. Collaborative filtering uses the similarity between users and items to generate recommendations. Content-based filtering uses the characteristics of the items to generate recommendations. Hybrid recommendation systems combine both approaches to provide more accurate and personalized recommendations.

Some common examples of recommendation systems include Netflix, Amazon, and YouTube. Netflix uses different methods to suggest movies and TV shows based on a user's viewing history and ratings. Amazon uses user data and item metadata to suggest products a user might be interested in. YouTube recommends videos based on a user's watch history and other data.

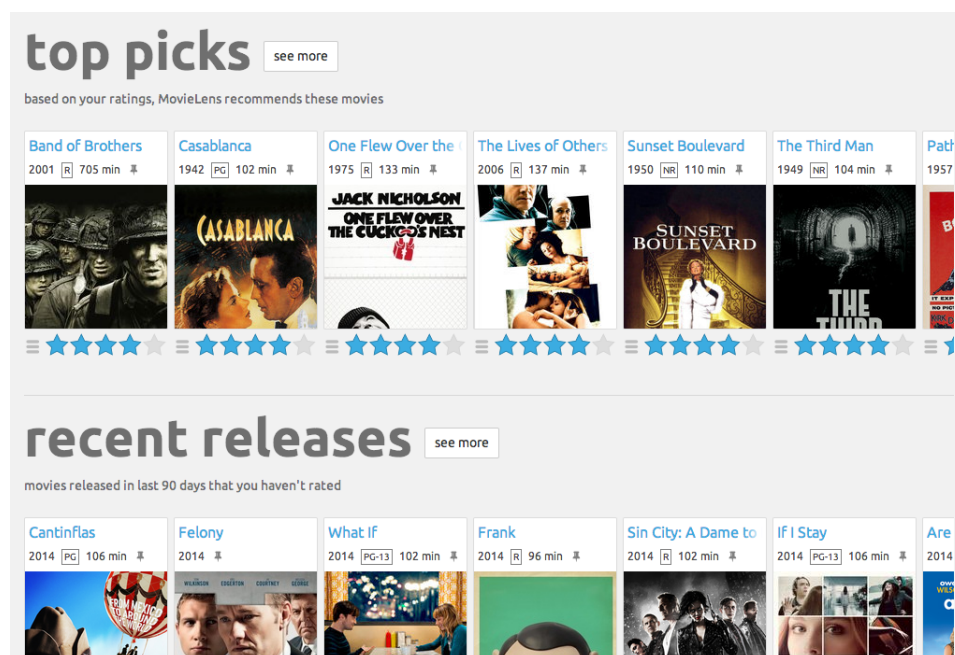


Figure 2.1: Screenshot from the Movie Lens [Ref[1]]

Figure 2.1 is a Screenshot from the Movie Lens [website](#), which showcases a recommendation system that utilizes technology to provide users with personalized movie recommendations. Analyzing users’ previous data and item metadata, the system generates a curated list of **top picks** that align with their interests. In addition, the platform also features a section for **recent releases** that utilizes user metadata to suggest newly released movies that align with their preferences. It is a unique system that genuinely understands its users and delivers results that keep them engaged and satisfied.

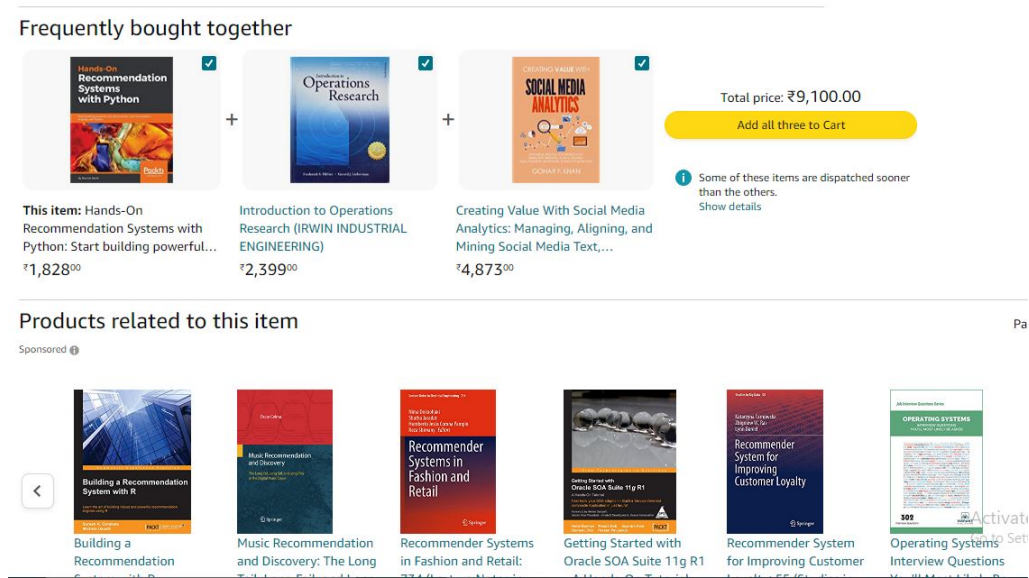


Figure 2.2: Screenshot from the Amazon [Ref[2]]

Figure 2.2 shows the screenshot of the Amazon [website](#) when users search for books. Where users can find a treasure trove of book recommendations.

At the very top, users can see the **Frequently bought together** section, where Amazon’s recommendation system uses advanced item metadata and transaction data to suggest items that are often purchased together. Moreover, just below that, the user can find the **Products related to this item** section, where the system offers an array of books that share similar metadata with the one currently being viewed.

2.3 Terminology

- **Cold-Start problem:** The cold-start problem refers to making recommendations for new users who have not yet provided any rating or preference data. Since collaborative filtering algorithms rely heavily on historical user behavior to make recommendations, it cannot be easy to generate accurate and relevant recommendations for new users without this information. The cold start problem can also apply to new items or products that have not yet been rated or evaluated by users, making their incorporation into the recommendation system challenging. Various techniques can be used to overcome the cold start problem, such as content-based filtering, hybrid recommendation systems, and asking for user preferences during the onboarding process.
- **Item information:** Item information refers to all the available data about a particular item in a recommendation system, such as its name, description, price, category, or other metadata.
- **User information:** User information refers to all the data available about the user in a recommendation system, such as their name, address, email address, demographic information, and other personal details.
- **Explicit information data:** Explicit information data refers to user feedback or ratings that are given directly by the user and reflect their explicit preferences or opinions about a particular item.
- **Implicit interaction data:** Implicit interaction data refers to user-item interactions that indicate interest or engagement with a particular item, such as views, clicks, or purchases. Unlike explicit ratings or feedback, implicit interactions do not directly reflect users' preferences or intentions but provide indirect signals about their interests.
- **Rating data:** Rating data refers to data containing ratings or feedback users give for items in a recommendation system. This data is typically used to train collaborative filtering algorithms, which rely on user-item interactions to make personalized recommendations.
- **Collaborative filtering algorithms:** Collaborative filtering algorithms are a type of recommendation system that make predictions based on the behavior or preferences of other users in the system. They use a user-item matrix to identify similar users or items and make recommendations based on those similarities.

Collaborative filtering can be very effective in generating personalized recommendations but requires sufficient user data and can suffer from the cold start problem.

- **Content-based filtering algorithms :** Content-based filtering algorithms are a type of recommendation system that use the attributes or features of items to make recommendations to users. They work by extracting relevant attributes or features from items, generating a user profile based on those attributes, and recommending items similar to the user profile. They do not require information about other users or their preferences, making them useful when there is limited or no user data.
- **Hybrid filtering algorithms:** Hybrid filtering algorithms are recommendation systems that combine the strengths of both collaborative and content-based filtering models. Collaborative filtering relies on user behavior data to generate recommendations, while content-based filtering relies on item features and attributes to make recommendations. By combining these two approaches, hybrid filtering algorithms can provide more accurate and personalized recommendations.

Types of Recommendation Systems

Recommendation algorithms[3] can be classified based on the recommendation type as personal or non-personal. Furthermore, recommendation algorithms can be categorized based on the type of data used for recommendation. If metadata is used, it is classified as content-based. If interaction data is used, it is referred to as collaborative filtering. Finally, if both data types are utilized, it is classified as a hybrid algorithm.

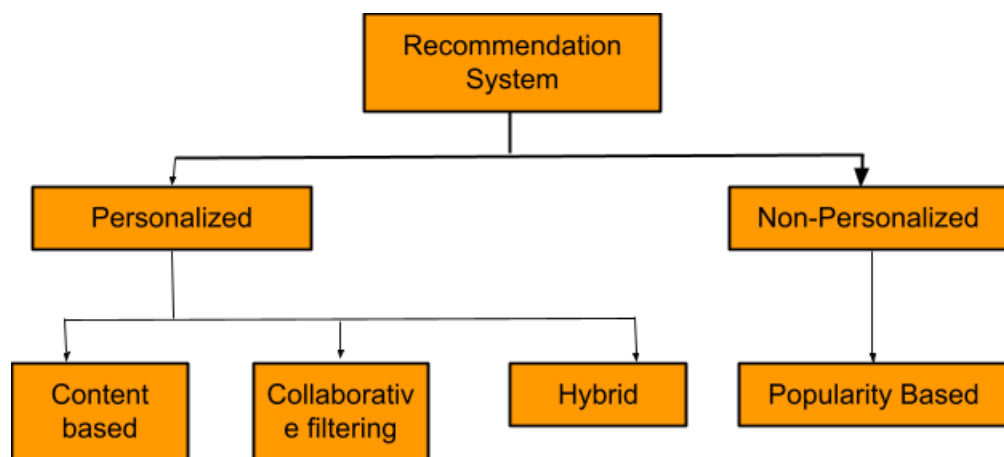


Figure 3.1: Types of Recommendation Systems

Figure 3.1 illustrates the classification of recommendation systems, where different types of recommendation systems are categorized into distinct classes.

3.1 Non - Personalized Recommendation System

Non-personalized recommendation systems make recommendations to all users based on the popularity of items without considering individual user preferences or behavior. These systems are often used when the user's data is limited or unavailable, such as for new users or in cases where users are yet to provide detailed ratings or feedback. The recommendations provided by non-personalized recommendation systems can be a good starting point for users in determining what they want. However, they may only be relevant or useful to some users, sometimes.

3.2 Personalized Recommendation System

Personalized recommender systems consider individual user preferences and behavior, such as past purchases, ratings, and user interactions, to provide personalized recommendations. These systems use various techniques, such as content-based filtering and collaborative filtering, to identify items likely to interest a particular user. By providing personalized recommendations, these systems can help users discover new items they might not have found otherwise, leading to increased engagement and satisfaction.

Content-based recommendation systems analyze the characteristics of items a user has liked and recommend similar items. In contrast, collaborative filtering recommendation systems look at user behavior patterns and recommend items that similar users have liked. Hybrid recommendation systems combine content-based and collaborative filtering approaches to provide more accurate and diverse recommendations.

3.2.1 Content-based Recommendation System

Content-based recommender systems are a sophisticated class of recommendation systems that rely on the user's purchase history and product metadata to create personalized recommendations. By carefully analyzing a user's past preferences and behaviors, these systems can determine their interests and preferences, allowing them to suggest items highly likely to appeal to the user. For example, suppose a user has purchased a book by a particular author or has shown an interest in a specific brand. In that case, content-based recommendation systems can leverage this information to recommend similar items with a high probability of being well-received by the user.



Figure 3.2: Content-based Recommendation System [Ref[4]]

Figure 3.2 highlights the power of a content-based recommendation system, which analyzes a user’s purchase or browsing history to suggest similar items of interest. In this case, the user had already read "Aristoi" by Walter Jon Williams, and based on this data, the system recommended "Angel Station," another book by the same author. This exemplifies how personalized recommendations can enhance the user experience and increase customer satisfaction.

3.2.2 Collaborative filtering (CF) Recommendation System

Collaborative filtering is a technique that utilizes the preferences and behavior of other users to provide personalized recommendations to a user. By analyzing ratings and user data, collaborative filtering can identify patterns and similarities between users and use this information to make accurate and relevant recommendations.

Collaborative filtering is divided into two main categories:

- **Memory based collaborative filtering :** the memory-based approach relies on the direct calculation of similarity between users or items
- **Model-based collaborative filtering :** the model-based approach uses techniques like matrix factorization to learn patterns and make predictions.

Collaborative filtering provides a highly personalized and practical recommendation experience to users.

Memory-based collaborative filtering recommendation system techniques analyze raw data such as rating, user and item data without preprocessing. These techniques are straightforward to implement and provide recommendations that are easy to understand and interpret. However, with these techniques, the system needs to make predictions over all the data each time, which can slow down the recommendation process. Despite this drawback, memory-based collaborative filtering remains a popular and effective approach for creating personalized recommendations.

Memory-based Collaborative filtering techniques are of two types:

- **User based Collaborative Filtering**

User-based collaborative filtering is a memory-based collaborative filtering technique that makes recommendations by finding other users with similar preferences to the target user and suggesting items similar to users who liked or rated highly. This approach utilizes the user-item rating matrix to find the similarity between users and recommend items based on the ratings given by similar users. User-based collaborative filtering is an intuitive approach to recommendation systems as it relies on the opinions and experiences of similar users to make recommendations for a given user.

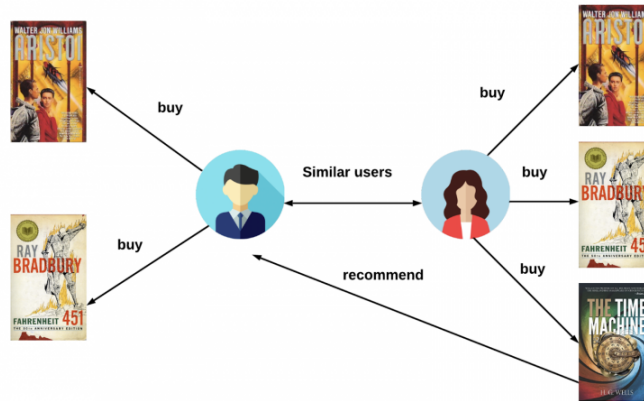


Figure 3.3: User-based Collaborative Filtering [Ref [4]]

As shown in figure 3.3, if user1 and user2 have similar purchase histories and user2 buys a new item, the system can also recommend that item to user1. In this case, user2 has bought three items, and 2 of those items have also been

purchased by user1. Based on this similarity, the system recommends item 3 to user 1, assuming that it might also interest them.

- **Item based Collaborative Filtering**

Item-based collaborative filtering is a collaborative filtering algorithm that recommends items to users based on the similarity or co-occurrence of the items themselves. In this approach, the algorithm first looks for items similar to the ones a user has already interacted with, then recommends those similar items to the User.

The item-based approach is generally faster and more scalable than the user-based approach because the similarity between items can be pre-computed and cached. In the user-based approach, user similarity must be computed at runtime. Additionally, item-based recommendations are more robust to new users and new items, as the similarity between items is generally more stable over time than between users.

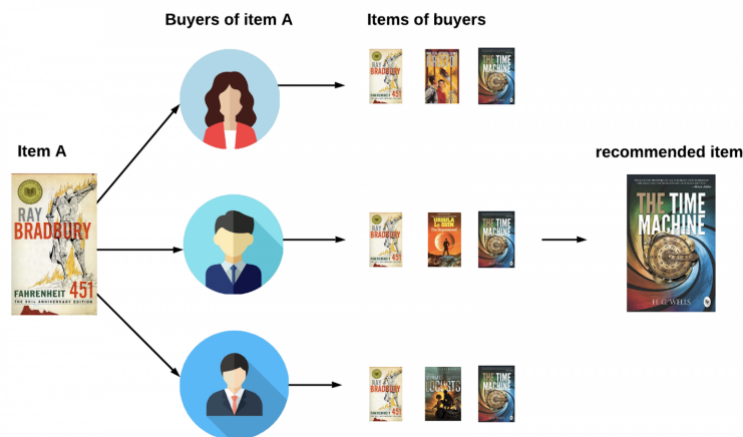


Figure 3.4: Item-based Collaborative Filtering [Ref[4]]

In figure 3.4, if item A is bought by user1, user2, and user3, and all three users have also shown interest in item C, then the item-based collaborative filtering system will recommend item C to any user who is looking at item A, based on the similarity between the two items.

Model-based collaborative filtering recommendation system algorithms are developed using machine learning algorithms. Instead of directly using the purchase or rating to compute cosine similarity, like memory-based CF, a model is created that

learns from the data and is used to make recommendations. This approach allows for faster processing of large datasets and can lead to better accuracy in recommendations.

One popular method in model-based CF is matrix factorization, where the user-item rating matrix is decomposed into two lower-dimensional matrices: a user matrix and an item matrix. The two matrices are multiplied together to reconstruct the original rating matrix. This allows for the prediction of missing ratings and the recommendation of new items to users.

Machine learning algorithms like decision trees, neural networks, and clustering can also be used in model-based CF to predict user preferences and suggest relevant items.

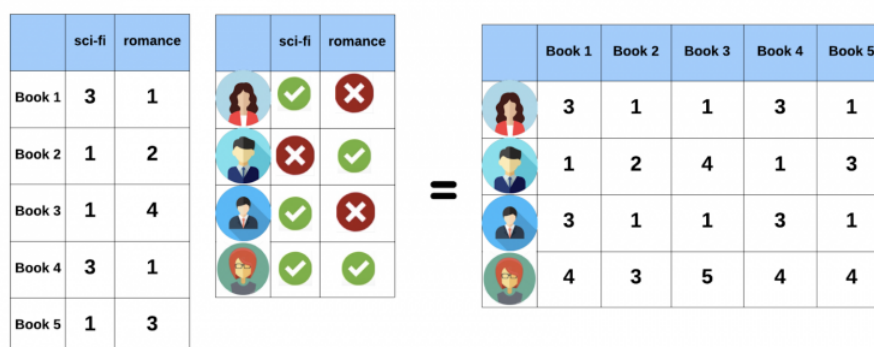


Figure 3.5: Matrix factorization [Ref[4]]

In figure 3.5, matrix factorization is applied to the data to extract representative vectors for both users and items. Each item is represented by its corresponding features, such as sci-fi and romance, and each user is represented by their preference for those features. Based on this representation, the system predicts a user's rating of a particular item and recommends it accordingly. This sophisticated approach enables personalized recommendations that reflect a user's unique tastes and preferences.

3.2.3 Hybrid Recommendation System

A hybrid recommendation system is a type of recommender system that combines multiple recommendation techniques to provide personalized recommendations to users. It represents a combination of different recommenders, leveraging the strengths of different recommendation approaches to overcome weaknesses and provide more accurate and diverse recommendations. A hybrid system can give better results than a single algorithm by combining several different recommenders. The goal of a hybrid system is

to provide more effective recommendations by considering multiple aspects of the user's preferences, such as their purchase history, browsing behavior, social media activity, and other factors.

The two main types of recommendation techniques used in hybrid systems are **Collaborative Filtering** and **Content-based Filtering**. These two techniques are combined differently in a hybrid system to provide more effective recommendations.

Hybrid systems can also integrate demographic, knowledge-based, and context-based filtering to augment the precision and comprehensiveness of recommendations.

- **Demographic Filtering:** Using demographic information such as age, gender, and location to make recommendations.
- **Knowledge-based Filtering:** Using a user's explicit preferences and requirements to make recommendations.
- **Context-based Filtering:** Using information about the user's current situation, such as time of day or location, to make recommendations.

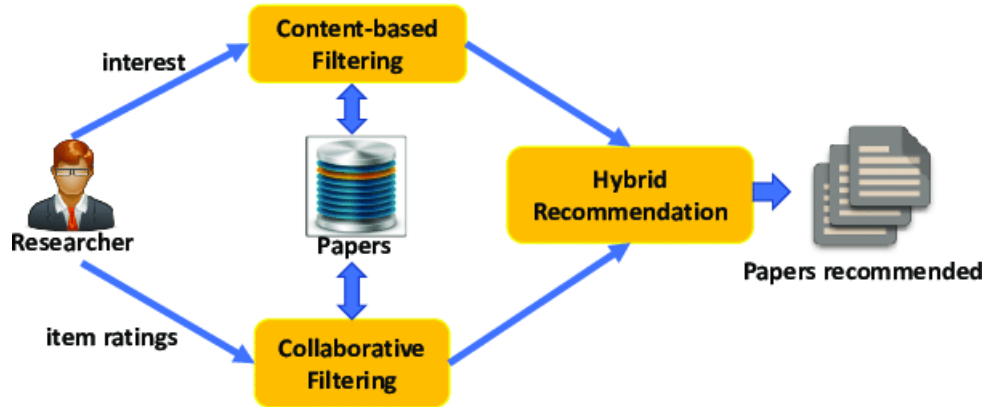


Figure 3.6: Hybrid Recommendation System [Ref [5]]

The given figure 3.6 uses content-based and collaborative filtering techniques to provide paper recommendations to a researcher. By combining these two techniques, the system can leverage the strengths of each approach to provide more accurate and diverse recommendations to the user. The content-based filtering technique analyzes the attributes of papers to recommend similar papers based on the User's past preferences. In contrast, the collaborative filtering technique uses historical behavior and preferences of similar researchers to identify patterns and similarities among them and recommend

popular papers among them. This hybrid approach can improve the accuracy and relevance of recommendations, thereby enhancing the researcher’s productivity and success in their field.

Conclusion

This chapter provides an overview of the different classification methods for recommendation algorithms. The classification is based on various factors, such as the level of personalization of the recommendation, the type of datasets utilized, and whether the recommendation system is based on memory or a model. A critical aspect of classification is the level of personalization of the recommendation. Personalized recommendation algorithms make recommendations based on an individual’s unique preferences and behavior, while non-personalized algorithms provide recommendations that are not tailored to any specific user. Another classification criterion is the type of datasets used in recommendation algorithms. Content-based algorithms utilize metadata such as item descriptions, while collaborative filtering algorithms utilize interaction data such as user ratings. Hybrid algorithms combine both types of data to make recommendations. Lastly, recommendation systems can be classified as either memory-based or model-based. Memory-based systems rely on comparing users or items to make recommendations, while model-based systems utilize machine learning algorithms to generate recommendations. Each system has advantages and disadvantages, and understanding the differences between them is essential in selecting the appropriate algorithm for a particular recommendation problem.

Chapter 4

Classifying Recommendation Systems

Recommendations can be used in a wide variety of situations, with a different number of users and items, and with new users and items being added. Another dimension is whether the user interacts with an item one time or multiple times. Different scenarios encountered in the real world are listed below.

- Large number of users, a small number of items, and one-time interaction:

Recommendation systems for movies and books need to provide diverse recommendations as users typically consume these items only once. Since the user-item interaction matrix for these domains is often dense, item-item similarity can be easily identified, and collaborative filtering algorithms can be used to make recommendations based on the ratings or preferences of similar users. However, it is important for recommendation systems to avoid simply recommending items that the user has already consumed and to take into account other factors such as genre, author, actor, director, and plot to provide more diverse and relevant recommendations. Content-based filtering can also be used to incorporate these additional factors into the recommendation process.

- Large number of users, a small number of items, and repeated interaction:

Recommendation systems for grocery or retail stores need to take into account the fact that users may repeatedly purchase the same items. In these domains, the user-item interaction matrix may be dense because there are many users and a limited number of items available. Collaborative filtering algorithms can still be used to identify user-user similarity and make recommendations based on the purchasing patterns of similar users. In addition, recommendation systems

can take into account factors such as brand, category, price, and promotions to recommend new items to users that they may be interested in.

- Small number of users, a large number of items, and one-time interaction:

News and video recommendation systems often face the challenge of having a constant number of users but new items are being added frequently. In this case, content-based recommendation systems can be more effective than collaborative filtering approaches because user-item interaction data is not available for the new items. Content-based systems can use features of the news or video content such as the topic, genre, language, keywords, or sentiment to recommend similar items to users. They can also use user preferences and feedback to refine the recommendations and provide a personalized experience. Additionally, hybrid recommendation systems that combine content-based and collaborative filtering techniques can be used to provide more diverse and accurate recommendations

- Small number of users, a large number of items, and repeated interaction:

Music recommendation systems face a similar challenge as news and video recommendation systems, where the number of users may be constant, but new items (songs, albums, artists) are continually being added. In such cases, content-based recommendation systems can be more effective than collaborative filtering approaches because user-item interaction data is not available for the new items. Content-based systems can use features such as the genre, artist, language, tempo, or mood of the music to recommend similar items to users. They can also use user preferences and feedback to refine the recommendations and provide a personalized experience. Hybrid recommendation systems that combine content-based and collaborative filtering techniques can also be used to provide more diverse and accurate recommendations.

- Large number of users, a large number of items, and one-time interaction:

E-commerce platforms like Amazon have a large number of users and items, and the user-item interaction matrix may be sparse because users typically only interact with items that they have purchased. Content-based recommendation systems may perform poorly in such scenarios because the diversity of products on the platform can make it challenging to identify meaningful features for item similarity. Additionally, the varying needs and preferences of users can make it difficult to generate relevant recommendations based solely on content. In these cases, memory-based collaborative filtering approaches can be more effective as they leverage the collective behavior of users to identify similarities and patterns in their purchasing behavior. Matrix factorization techniques such as singular

value decomposition (SVD) and non-negative matrix factorization (NMF) can also be used to extract latent features from the user-item interaction matrix and improve the accuracy of recommendations.

- Large number of users, a large number of items, and repeated interaction:

Fashion stores like H&M have a large number of users and items, and the user-item interaction matrix may be sparse because users typically only interact with items that they have purchased. In such scenarios, hybrid recommendation systems that combine collaborative filtering and content-based approaches can be more effective. Collaborative filtering can be used to identify user preferences and generate recommendations based on the behavior of similar users, while content-based filtering can be used to incorporate item features such as color, style, fabric, and brand to provide more personalized and diverse recommendations. Hybrid approaches can also address the cold-start problem where new items or new users do not have sufficient interaction data by leveraging item metadata and user demographics to provide relevant recommendations. Finally, deep learning techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) can also be used to extract features from images and text data to improve the accuracy of recommendations in fashion stores.

Evaluation metrics

Evaluation metrics [6] is essential to assess the performance of a recommendation system. The evaluation metrics vary based on the type of recommendation system and the task at hand. Here are some standard evaluation metrics for recommendation systems.

5.1 Root Mean Squared Error (RMSE)

Root Mean Squared Error (RMSE) [7] is a way of measuring how well a recommendation system works. It measures the difference between users' actual ratings and the rating predicted by the recommendation system. The lower the RMSE value, the better recommendations that the user enjoys. If the RMSE value is low, the system suggests things the user might like very well.

$$RMSE = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}$$

where, r_{ui} is the actual rating, \hat{r}_{ui} is the predicted rating and R is the set of all ratings.

5.2 Precision at K

Precision at K [8] is a critical way to measure how good a recommendation system is at suggesting things a user will like. It looks at how many top-K recommendations

match the user’s preferences. The higher the precision at K, the better the system gives personalized and accurate recommendations matching user preferences. Studying precision at different K values helps to improve the system and find the best K value for maximum effectiveness.

$$Prec(R)_k = \frac{|r \in R : r \leq K|}{K}$$

Where R is a set of relevant items

5.3 Recall at K

Recall at K [8] is a way to measure how good a recommendation system is at suggesting things a user will like. It measures how many of the user’s actual preferences are recommended by the system in the top K items. If the recall at K value is high, the system is good at identifying all the relevant items, not just the top few. This is important because it helps the system give better recommendations. Recall at K is also helpful because it lets us compare the system’s performance at different values of K. This helps us find the best value of K for getting the most relevant recommendations.

Overall, recall at K is a critical way to evaluate recommendation systems. It helps them give better-personalized recommendations, which leads to happier users.

$$recall(R)_k = \frac{|r \in R : r \leq K|}{|R|}$$

Where R is a set of relevant items

5.4 Area under the curve (AUC)

AUC [9] is a widely used evaluation metric for recommendation systems, which measures the ability of the system to distinguish between relevant and irrelevant items accurately. It does this by calculating the proportion of relevant item and non-relevant item pairs where the recommendation algorithm ranks the relevant item. A higher AUC score indicates a better recommendation system that can predict which items will be relevant to the user more accurately. By optimizing AUC, recommendation systems

can provide more personalized and relevant recommendations, improving the overall user experience and satisfaction.

$$AUC(R)_n = \frac{1}{|R|(n - |R|)} \sum_{r \in R} \sum_{r' \in (\{1,2,\dots,n\} \setminus R)} \delta(r < r')$$

$$\delta(r < r') = \begin{cases} 1 & \text{if Rank of Relevant item is higher then irrelavent item} \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

5.5 Accuracy

Accuracy is another widely used evaluation metric for recommendation systems. It measures the system's ability to correctly predict or classify user preferences or behaviors, such as whether a user will like or dislike a particular item.

In the context of recommendation systems, accuracy measures how well the system can predict the correct outcome, such as whether a user will interact with or purchase a recommended item. It is calculated as the ratio of correctly predicted observations to total predicted observations.

Accuracy is a crucial factor to consider when developing and evaluating recommendation systems, as it directly impacts the quality of recommendations and the overall user satisfaction with the system. By optimizing accuracy, recommendation systems can provide more relevant and personalized recommendations that accurately reflect the preferences and needs of each user, thereby improving the overall user experience and engagement.

Chapter 6

Models

Implementing recommendation system algorithms requires a powerful, flexible programming language that can easily handle complex data processing and machine learning tasks. Python, being one of the most popular programming languages, is a natural choice for this purpose. Its extensive libraries and frameworks, such as NumPy, Pandas, Scikit-learn, LightFM, TensorFlow, PyTorch and Keras, provide a comprehensive range of tools and techniques for developing highly efficient and accurate recommendation systems.

Python's rich ecosystem of data science and machine learning communities has made it a go-to choice for many data scientists and researchers in recommendation systems. Moreover, the vast online tutorials, blogs, and documentation has made it easier for developers to learn and implement new techniques in their projects.

Here are some of the recommendation algorithms implemented in this project.

6.1 Hybrid Model

A hybrid recommendation system[5] combines multiple recommendation techniques to provide more accurate and diverse recommendations. Content-based and collaborative filtering are two popular approaches used in building hybrid recommendation systems.

6.1.1 Matrix factorization

Matrix factorization[10] is a technique used in collaborative filtering-based recommendation systems. In a hybrid recommendation system, matrix factorization can combine both collaborative and content-based filtering techniques. Here is a mathematical explanation of matrix factorization in a hybrid recommendation system:

Let R be an $n \times m$ matrix, where n is the number of users and m is the number of items. The matrix R contains each user's ratings for each item. In a collaborative filtering approach, the goal is to factorize the matrix R into two matrices, P and Q , such that:

$$R \approx P \times Q$$

Where P is an $n \times k$ matrix, and Q is a $k \times m$ matrix. Here, k is the number of latent factors that are used to represent the preferences of users and the attributes of items. The matrix P represents the user preferences for the k latent factors, and the matrix Q represents the attributes of items for the k latent factors.

In a content-based filtering approach, the goal is to represent the attributes of each item as a vector. Let X be an $m \times p$ matrix, where p is the number of attributes used to describe each item. Each row of the matrix X represents the attribute vector of an item.

A hybrid matrix factorization approach combines these two approaches. This approach creates a new matrix Y by concatenating the matrix R and matrix X vertically. Let Y be an $(n + p) \times m$ matrix, where the first n rows represent the ratings of users for items, and the remaining p rows represent the attributes of each item.

Then factorize the matrix Y into two matrices, P and Q' , such that:

$$Y \approx P \times Q'$$

The matrix P represents the user preferences for the k latent factors, and the matrix Q' represents the combined attributes of items for the k latent factors.

To obtain the recommendation for a user u for an item i , the predicted rating can be calculated by using the dot product of the user preference vector and the item attribute vector, i.e.,

$$\text{predicted rating}(u, i) = P[u, :] \times Q'[i, :]^T$$

where $P[u, :]$ represents the k -dimensional preference vector for user u , and $Q'[i, :]^T$ represents the k -dimensional attribute vector for item i .

Using matrix factorization in a hybrid recommendation system combines the strengths of collaborative filtering and content-based filtering techniques to provide more accurate and diverse recommendations.

6.1.2 LightFM

LightFM [11] is a Python library commonly used to implement hybrid recommendation systems. It implements a hybrid matrix factorization model that combines content-based and collaborative filtering techniques for the recommendation. LightFM can handle both implicit and explicit feedback. One of the unique features of LightFM is its ability to incorporate both item and user metadata into the matrix factorization algorithms, which significantly enhances the system's ability to generalize to new items and new users. LightFM can recommend new items to users with similar preferences using item features. In contrast, user features help to identify new users who are likely to be interested in certain items. This flexibility allows LightFM to provide personalized and diverse recommendations that meet the specific needs of users.

In order to build a recommendation system using LightFM, three types of data are needed: **Rating data, User features, Item features**. While rating data is typically required for training the recommendation model, user and item features are optional but can improve the quality of the recommendations. If user or item features are unavailable, the model can still be trained using only the rating data.

LightFM supports several loss functions and learning algorithms that can be used for training the recommendation model.

- **The three loss functions supported by LightFM are:**

- **WARP (Weighted Approximate-Rank Pairwise):** This is a loss function for implicit feedback recommendation problems. WARP aims to optimize the rank of positive items relative to negative items, with higher weights placed on more difficult ranking pairs.

- **WARP-KOS (Weighted Approximate-Rank Pairwise with K-OS):** This is a loss function for explicit feedback recommendation problems. This is a variant of the WARP loss function that incorporates the idea of K-OS (K-Order Statistic Loss), which introduces a new set of ranking pairs for each training example and applies a softmax function to the scores to calculate the probability of a positive item being ranked higher than a negative item.
- **BPR (Bayesian Personalized Ranking):** This is another loss function for implicit feedback recommendation problems. BPR aims to maximize the difference between each user’s predicted scores of positive and negative items.
- **The two learning algorithms supported by LightFM are:**
 - **Adagrad (Adaptive Gradient Algorithm):** This optimization algorithm adapts the learning rate for each parameter based on historical gradient information.
 - **Adadelta (Adaptive Delta Algorithm):** This is another optimization algorithm that adapts the learning rate based on the second moment of the gradient rather than the historical gradient information.

Choosing the appropriate loss function and learning algorithm depends on the specific characteristics of the recommendation problem, such as the type of feedback data and the desired trade-offs between accuracy and training time.

6.2 Generalized Matrix Factorization (GMF)

Generalized Matrix Factorization (GMF) represents a recommendation algorithm that employs matrix factorization to model user-item interactions, the goal is to factorize a rating matrix R into two low-dimensional matrices U and V , such that their product UV^T approximates R . Here, U is an $m \times k$ matrix that represents k -dimensional embeddings for m users, and V is an $n \times k$ matrix that represents k -dimensional embeddings for n items. The dot product of the i^{th} row of U and the j^{th} row of V^T gives an estimate of the rating that the user i would give to item j .

To learn the embeddings, GMF minimizes a loss function that measures the difference between the predicted ratings and the actual ratings in the training data. A common loss function used in GMF is the mean squared error (MSE), which is defined as:

$$L = \frac{\sum_{(i,j) \in D} (R_{i,j} - U_i * V_j^T)^2}{N}$$

where N is the number of observed ratings in the training data D , and $R_{i,j}$ is the actual rating that user i gives to item j .

To minimize the loss function, GMF typically uses stochastic gradient descent (SGD) or some other optimization algorithm. The gradients of the loss function with respect to U and V can be computed using backpropagation.

One key innovation of GMF is to use element-wise multiplication (i.e., Hadamard product) of the user and item embeddings to capture their interactions, instead of concatenation or addition. The element-wise multiplication can be written as:

$$f(U_i, V_j) = \sum_{k=1}^k U_{i,k} * V_{j,k}$$

where $*$ denotes the element-wise multiplication. The output $f(U_i, V_j)$ is a scalar that represents the estimated rating of user i for item j . By training the embeddings to minimize the difference between $f(U_i, V_j)$ and $R_{i,j}$, GMF can learn to recommend items that are likely to be preferred by users.

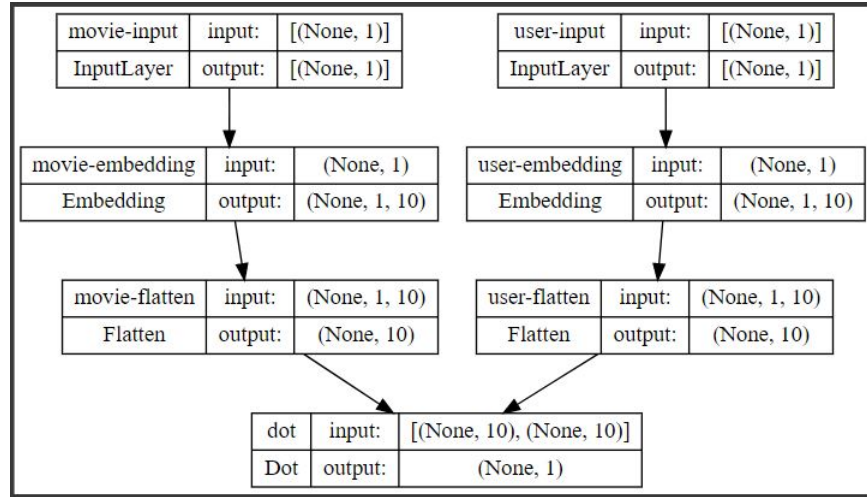


Figure 6.1: Architecture of GMF

The architecture of GMF using Tensor Flow, depicted in Figure 6.1, involves embedding both the item and user into a size-10 embedding and utilizing these embeddings to predict the rating. Mean squared error loss with Adam optimizer is used to train the model.

6.3 Neural Collaborative Filtering (NCF)

Neural Collaborative Filtering (NCF)[12] is a recommendation algorithm that harnesses the power of neural networks to model user-item interactions. By embedding users and items into a low-dimensional space, NCF can capture complex patterns and learn the subtle nuances that influence user preferences.

The architecture of NCF is built on a two-part foundation consisting of the embedding layer and the second part consisting of the MLP layer. The embedding layer facilitates learning low-dimensional representations of users and items, while the MLP layer models the intricate and nonlinear user-item interactions. The joint training of these two layers allows NCF to minimize a loss function, which measures the difference between predicted and actual ratings in the training dataset.

NCF captures explicit and implicit feedback and learns from ratings and interactions, such as clicks and purchases. Moreover, NCF can handle both, sparse and dense datasets.

NCF Architecture

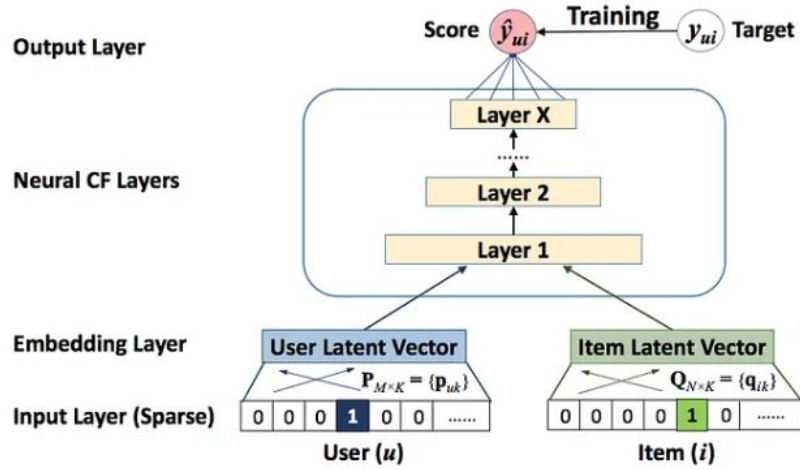


Figure 6.2: Neural Collaborative Filtering [Ref [12]]

In Figure 6.2 architecture of NCF is shown, a recommendation algorithm that is comprises of two distinct and equally essential parts. The first, the Embedding Layers, facilitates the learning of low-dimensional representations of users and items, while the second, the Neural CF Layers, models the complex user-item interactions.

6.4 Neural Matrix Factorization(NeuMF)

NeuMF[12] is a deep learning model which combines GMF (Generalized Matrix Factorization) and MLP (Multi-Layer Perceptron) to improve the accuracy of recommendations in recommender systems. By combining the strengths of matrix factorization and neural networks, NeuMF can capture nonlinear patterns in data and incorporate additional user and item features beyond latent factors.

Please refer to section 6.2 for GMF and section 6.3 for NCF.

NeuMF is a recommendation algorithm that overcomes the limitations of GMF. By incorporating a neural network component, NeuMF can capture nonlinear patterns in data and extract additional user and item features beyond latent factors. This is achieved through a unique architecture combining a traditional matrix factorization model with a powerful multi-layer perceptron (MLP) neural network model, both concatenated and fed into the NeuMF layer to produce accurate and reliable recommendations.

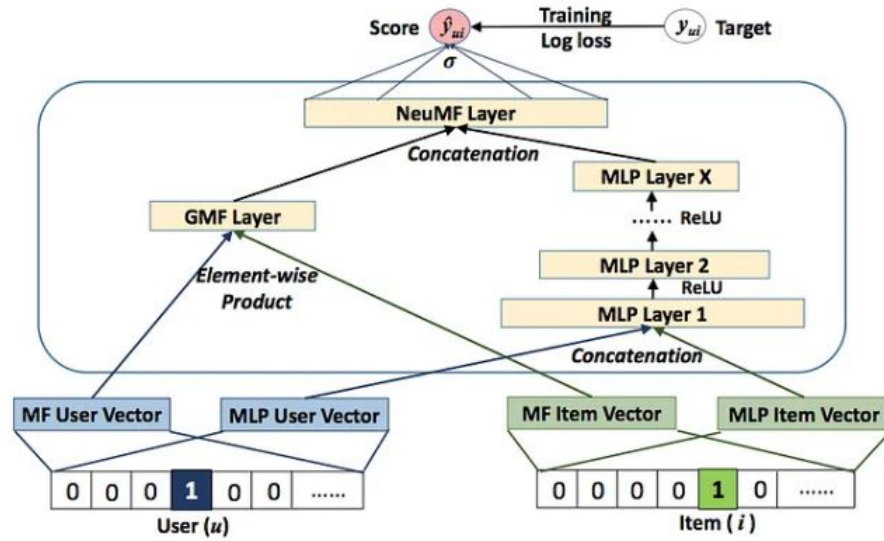


Figure 6.3: NeuMF Architecture [Ref [12]]

In Figure 6.3. the architecture of NeuMF consists of two parts: the GMF and MLP components. The GMF component takes the user and item embeddings generated by the matrix factorization (MF) part and feeds them into a GMF layer, which produces

a low-rank interaction matrix. The MLP component takes the same user and item embeddings generated by the MLP part and feeds them into a series of fully connected MLP layers. The outputs of the GMF and MLP components are concatenated and passed through a final NeuMF layer to generate the final prediction. This architecture allows NeuMF to capture linear and nonlinear relationships between users and items, improving recommendation accuracy.

Two different architectures for the NeuMF layer:

- 1. Single fully connected layer:** In this architecture, the concatenated output of the GMF and MLP components are fed into a single fully connected layer in the NeuMF layer. This layer has a nonlinear activation function (ReLU is used) and produces the final recommendation scores.
- 2. Three fully connected layers:** In this architecture, the concatenated output of the GMF and MLP components is fed into three fully connected layers in the NeuMF layer. Each layer has a nonlinear activation function (ReLU is used), and the outputs are passed through dropout layers to prevent overfitting. The final layer produces the recommendation scores.

6.5 Alternating Least Squares (ALS)

Alternating Least Squares (ALS)[\[13\]](#) is a recommendation algorithm that uses matrix factorization to model user-item interactions. A collaborative filtering algorithm seeks to capture the underlying latent factors that influence user preferences. In ALS, a user-item interaction matrix R is decomposed into two lower-dimensional matrices: a user matrix U and an item matrix V .

Let R be an interaction matrix and U and V are user and item latent factors respectively.

$$R \approx U \times V$$

Each user matrix and item matrix element represents a score or weight for a particular latent factor. To learn the weights, ALS minimizes a loss function that measures the difference between the predicted ratings and the actual ratings in the training data. The loss function used in ALS is usually the sum of squared errors (SSE), which is defined as:

$$L = \sum_{(i,j) \in D} (R_{i,j} - U_i \cdot V_j^T)^2 + \lambda(\|U\|^2 + \|V\|^2)$$

where D is the set of observed ratings in the training data, λ is a regularization parameter, and $\|U\|$ and $\|V\|$ are the L2 norms of the rows of U and V , respectively. The regularization term helps to prevent overfitting by penalizing large values in the embedding matrices.

To minimize the loss function, ALS alternates between fixing U and optimizing V , and fixing V and optimizing U . When fixing U and optimizing V , treat U as a constant and solve for V by minimizing the loss function with respect to V . This can be done by taking the derivative of the loss function with respect to V and setting it to zero:

$$V = (U^T U + \lambda I)^{-1} U^T R$$

where I is the identity matrix of size $k \times k$. When fixing V and optimizing U , treat V as a constant and solve for U by minimizing the loss function with respect to U . This can be done by taking the derivative of the loss function with respect to U and setting it to zero:

$$U = (V^T V + \lambda I)^{-1} V^T R^T$$

where R^T is the transpose of R .

The process of alternating between fixing U and optimizing V , and fixing V and optimizing U , is repeated until convergence. The convergence criterion can be based on the decrease in the loss function or the change in the embedding matrices. ALS has been shown to be effective in recommendation systems, especially when the rating matrix is sparse and the data has a large number of users and items. ALS is a relatively simple and computationally efficient algorithm that applies to many recommendation tasks in various domains, including e-commerce, music, and video streaming.

6.6 Deep Walk

DeepWalk[14] is a graph embedding technique that learns low-dimensional representations, or embeddings, of nodes in a graph. These embeddings can be used for various downstream tasks such as link prediction, node classification and recommendation.

DeepWalk is used to learn embeddings for items in the dataset in the context of buy-together data. Each node in the graph represents an item, and the edges between nodes represent the frequency of co-occurrence of those items in the same transaction.

Here is a step-by-step explanation of how to use DeepWalk on this type of data:

- **Construct the Graph:** First, construct a graph representation of the buy-together data. A node in the graph represents each item in the dataset, and the edges between nodes are weighted by the frequency of co-occurrence of those items in the same transaction. For example, if item A and item B were bought together 10 times, there would be an edge between the nodes representing those items with a weight of 10.
- **Generate Random Walks:** Next, generate random walks on the graph. A random walk is a sequence of nodes obtained by traversing the graph, starting at a random node and following a sequence of edges until a desired length is reached. Repeat this process many times to generate multiple random walks of fixed length from each node in the graph.
- **Learn Embeddings:** Then, use the sequences of nodes obtained from the random walks to learn embeddings for the nodes in the graph. Specifically, the skip-gram model is used to learn the embeddings. The skip-gram model is trained to predict the probability of observing a node in a random walk given its neighboring nodes. The embeddings learned by the skip-gram model represent the nodes in a low-dimensional space, so nodes that frequently co-occur in random walks are closer together in the embedding space.
- **Use Embeddings for Recommendation:** The learned embeddings are used for recommendation tasks. One common approach is to compute the similarity between the embeddings of a set of items and the embeddings of all other items in the graph and recommend the items with the highest similarity scores. Another approach is to use embeddings as features in a machine learning model, such as a neural network, to predict the likelihood that a set of items will be bought together.

Overall, DeepWalk is a technique for learning node embeddings in graphs and can be applied to buy-together data to learn embeddings for items and make recommendations based on those embeddings.

Chapter 7

Datasets and Implementation

There are two main types of data used in recommendation systems:

- **Explicit data:** This type of data is obtained directly from the user and includes ratings, reviews, and feedback the user provides about a product or service.
- **Implicit data:** This type of data is obtained indirectly from the user and includes user behavior such as purchase history, browsing history, search history, and click-through rates.

The value of K for Precision@K and Recall@K is set to 20 for all models and datasets.

7.1 Movie Lens Dataset

The Movie-Lens dataset is used for building and evaluating recommendation systems. It contains movie ratings and other metadata collected from users of the Movie-Lens website, a movie recommendation service. The dataset[15] has several versions with varying numbers of users, movies, and ratings. The most commonly used version, Movie-Lens 100K, contains 100,000 ratings on a scale of 1 to 5 from 610 users on 9724 movies. This data is used for training and testing collaborative filtering and other recommendation algorithms.

Exploratory data analysis

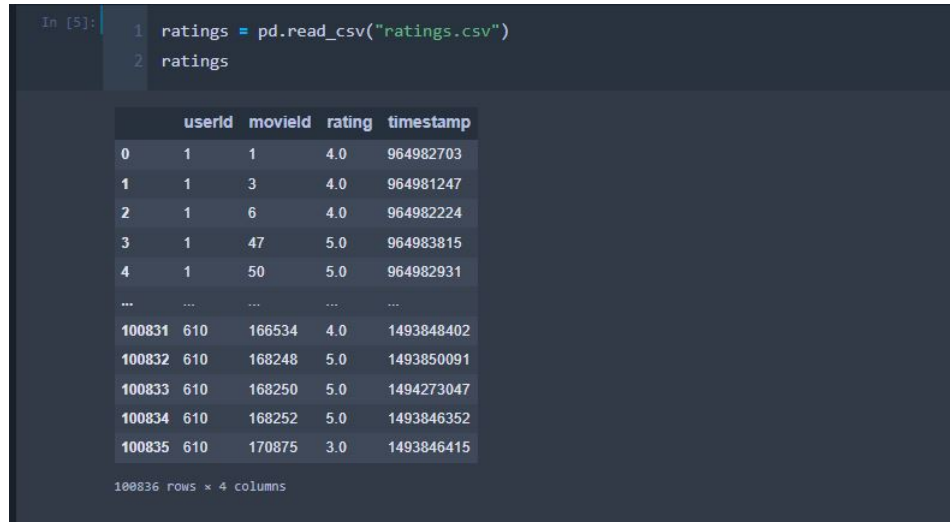


Figure 7.1: Movie-Lens rating dataset



Figure 7.2: Movie-Lens movies dataset

There are **100k** ratings Figure 7.1 given by **610** users for **9724** movies Figure 7.2. There are **3446** movies with only one rating, meaning that they are watched by only one user. On an average each movie is watched by **10** users and each user watched **160** movies.

Figure 7.3 shows the histogram of the number of users who watched a particular movie.

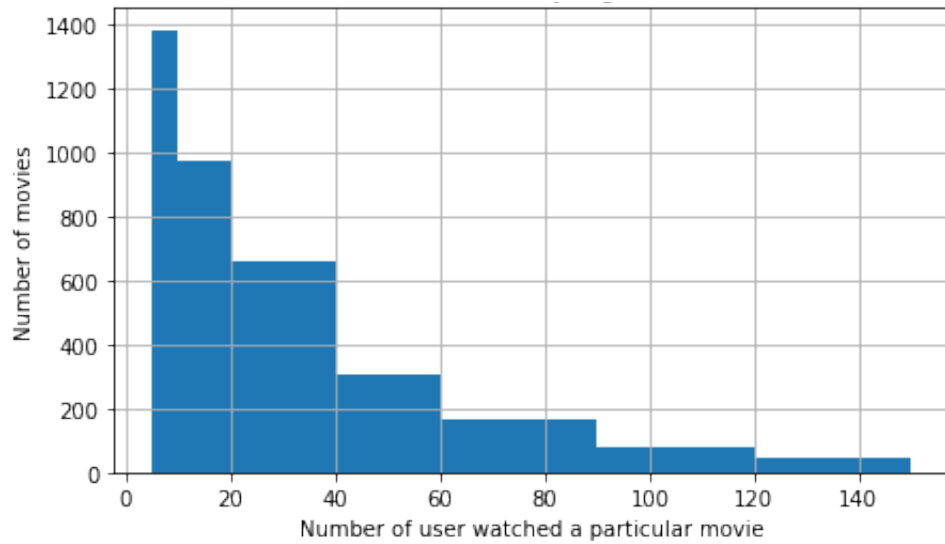


Figure 7.3: Histogram of number of users who watched a particular movie

Figure 7.4 shows the histogram of the number of movies watched by a particular user.

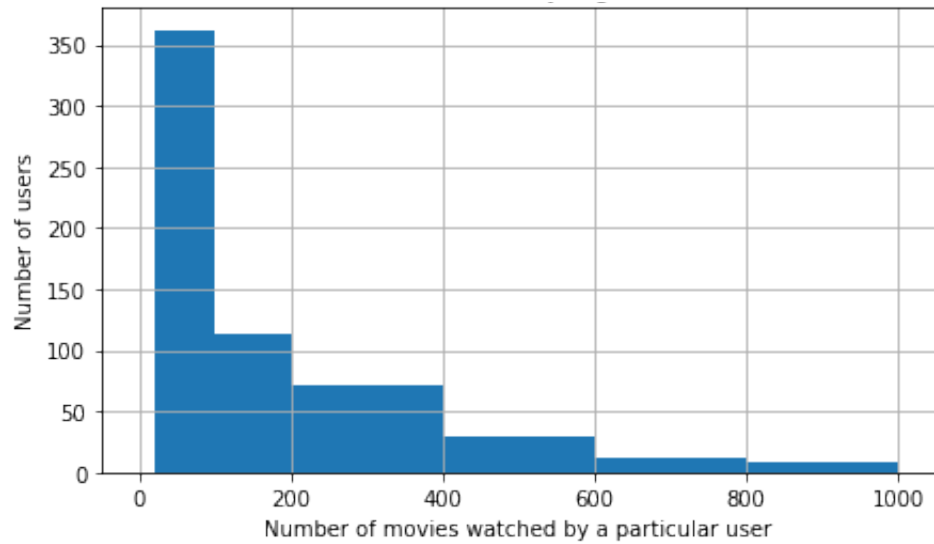


Figure 7.4: Histogram of number of movies watched by a particular user

Figure 7.5 shows the number of movies for each genre.

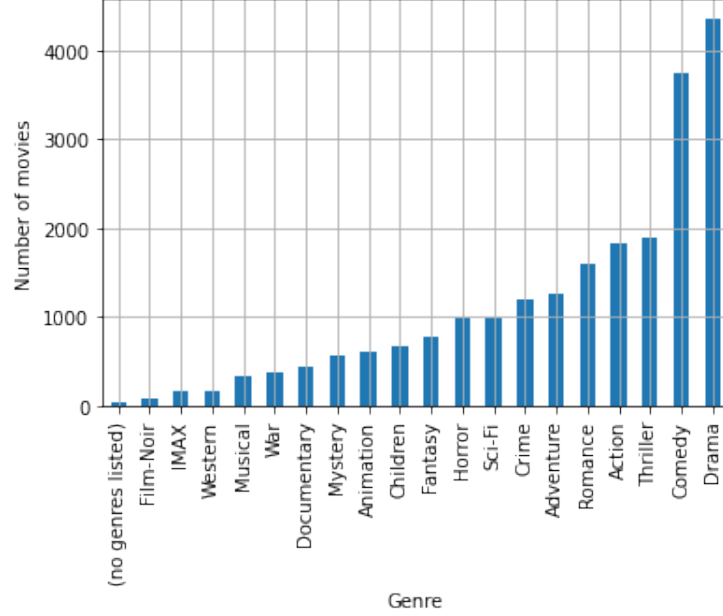


Figure 7.5: Number of movies for each genre

Implementation and results of algorithms

The dataset is split is 80:20 for train and testing respectively. The dataset is divided using the time and date of the rating. first 80% of data is used for training and the remaining 20% for testing.

- **Hybrid model**

For the Movie-Lens dataset results using LightFM are shown in Table 7.1.

Table 7.1: Result of LightFm on movie lens dataset

Learning	Loss Func	Train Time (sec)	Rec Time (sec)	AUC	Pre@K	Rec@K
adagrad	WARP	12.858	0.004	0.780	0.415	0.459
	WARP_KOS	13.861	0.005	0.784	0.400	0.472
	BPR	16.494	0.009	0.672	0.352	0.352
adadelata	WARP	19.614	0.004	0.807	0.649	0.509
	WARP_KOS	18.770	0.006	0.821	0.659	0.517
	BPR	26.211	0.008	0.752	0.401	0.385

Upon analyzing the LightFM recommendation system, it has become clear that

the choice of the learning model impacts training time and performance. Specifically, it has been observed that while Adadelta generally takes more time to train compared to Adagrad across all loss functions, it gives good Precision@K for Movie-Lens dataset. Moreover, the WARP_KOS loss function gives good results for the Movie-Lens dataset.

- **Generalized matrix factorization (GMF)**

For the Movie-Lens dataset results using GMF are shown in Table 7.2.

Table 7.2: Result of GMF on movie lens dataset

Iter		Train Time(sec)	Rec Time(sec)	AUC	Pre@K	Rec@K	RMSE
10	Normal weights	144	1.3	0.673	0.541	0.424	1.613
10	Non-neg weights	178	0.7	0.674	0.552	0.415	1.617
20	Normal weights	283	0.7	0.723	0.571	0.441	1.483
20	Non-neg weights	383	0.7	0.734	0.575	0.450	1.576

- **NCF**

For the Movie-Lens dataset results using NCF are shown in Table 7.3.

Table 7.3: Result of NCF on movie lens dataset

Iter	Train Time(sec)	Rec Time(sec)	AUC	Pre@K	Rec@K	RMSE
10	248	1.39	0.786	0.619	0.561	1.360
20	504	1.90	0.807	0.623	0.596	1.343

- **NueMF**

For the Movie-Lens dataset results using NueMF are shown in Table 7.4.

Table 7.4: Result of NueMF on movie lens dataset

Iter	FC layer	Train Time(sec)	Rec Time(sec)	AUC	Pre@20	Rec@20	RMSE
10	1	384	0.90	0.806	0.632	0.612	1.264
20	1	685	1.31	0.836	0.636	0.617	1.215
10	3	445	2.68	0.833	0.653	0.628	1.174
20	3	805	1.22	0.879	0.656	0.639	1.174

In NeuMF architecture, adding more fully connected Layers in NeuMF can improve the performance of the Movie-Lens dataset. Specifically, adding three fully

connected layers in the NeuMF layer improves the AUC (area under the ROC curve) compared to using only one fully connected layer. Increasing the number of training iterations from 10 to 20 further improves the performance of NeuMF on the Movie-Lens dataset. This indicates that NeuMF benefits from longer training times and can continue to improve with more iterations up to a certain point.

Conclusion

Typical movie recommendations are characterized by a large number of users, a small number of items (movies), and one-time interaction since a user typically watches a movie only once. Collaborative Filtering algorithms are often used to generate personalized recommendations in such scenarios where the user-item interaction matrix is sparse. Furthermore, when a dataset has a higher number of users and a smaller number of items, coupled with rating and item data, item-item collaborative filtering algorithms with item features can be effective in generating accurate and relevant recommendations. These algorithms leverage the collective intelligence of the item-item collaborative filtering and item features to identify similar patterns and preferences, which can be used to provide accurate recommendations to the users.

Overall, item-item collaborative filtering algorithms can be highly effective in recommending movies or books to users based on their past interactions with the dataset. The MovieLens dataset provides a good example of how these algorithms can be used to generate personalized recommendations.

7.2 H & M Fashion Dataset

The H & M fashion dataset[16] has transactions and article data. The transactions data set captures information on customer purchases, including purchase history, product preferences, and purchasing patterns. This data can be used to build recommendation models tailored to each customer’s unique needs and preferences, resulting in more relevant and personalized recommendations. Meanwhile, the article data set component provides detailed information on product attributes such as style, color, and size, which can be used to develop more sophisticated product matching algorithms that better reflect the nuanced and complex nature of fashion retail. Collectively, The H & M fashion dataset is a resource for those seeking to develop recommendation systems in the fashion domain, allowing retailers to optimize their offerings and deliver a more satisfying shopping experience to their customers.

Exploratory data analysis

```
In [17]: 1 item = pd.read_csv("articles.csv")
          2 item.head()
```

	article_id	product_code	prod_name	product_type_no	product_type_name	product_group_name	graphical_appearance_no
0	108775015	108775	Strap top	253	Vest top	Garment Upper body	1010016
1	108775044	108775	Strap top	253	Vest top	Garment Upper body	1010016
2	108775051	108775	Strap top (1)	253	Vest top	Garment Upper body	1010017
3	110065001	110065	OP T-shirt (ldro)	306	Bra	Underwear	1010016
4	110065002	110065	OP T-shirt (ldro)	306	Bra	Underwear	1010016

Figure 7.6: H & M fashion article data set

```
In [29]: 1 rating = pd.read_csv("transactions_train.csv")
          2 rating
```

	t_dat	customer_id	article_id	price	sales_channel_id
0	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2
1	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	541518023	0.030492	2
2	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	505221004	0.015237	2
3	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687003	0.016932	2
4	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687004	0.016932	2
...
31788319	2020-09-22	fff2282977442e327b45d8c89afde25617d00124d0f999...	929511001	0.059305	2
31788320	2020-09-22	fff2282977442e327b45d8c89afde25617d00124d0f999...	891322004	0.042356	2
31788321	2020-09-22	fff380805474b287b05cb2a7507b9a013482f7dd0bce0e...	918325001	0.043203	1
31788322	2020-09-22	fff4d3a8b1f3b60af93e78c30a7cb4cf75edaf2590d3e5...	833459002	0.006763	1
31788323	2020-09-22	ffef3b6b73545df065b521e19f64bf6fe93bfd450ab20...	898573003	0.033881	2

Figure 7.7: H & M fashion transaction data set

There are **91996** transactions shown in Figure 7.7 by **24168** unique costumers, purchasing **5550** unique articles shown in Figure 7.6. There are **7537** customers who bought only one article. There are **1293** articles bought by only one customer. On an average, each article is bought by **3.8** users and each user bought **16** articles.

Figure 7.8: shows the histogram of the number of articles purchased by a particular customer.

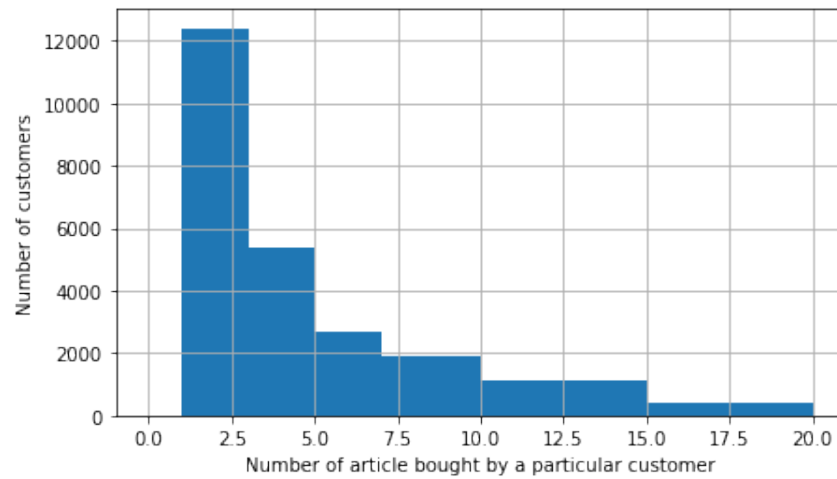


Figure 7.8: Histogram of number of articles purchased by a particular customer

Figure 7.9 shows the histogram of the number of customers who bought a particular article.

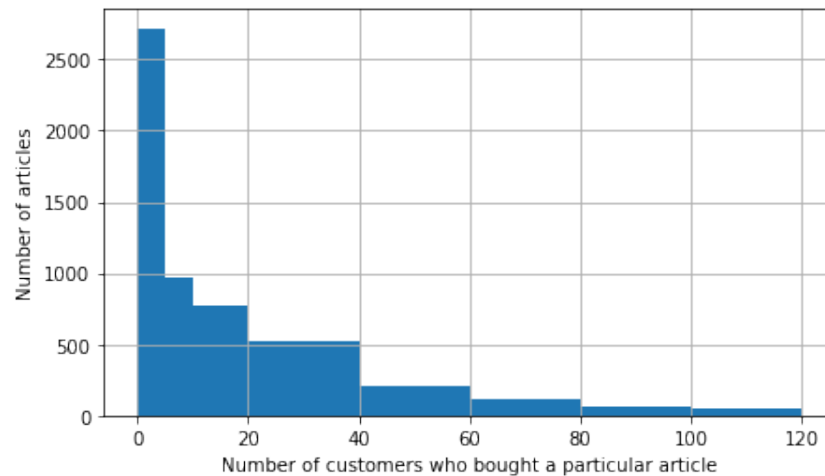


Figure 7.9: Histogram of the number of customers who bought a particular article

Implementation and results of algorithms

The dataset is split is 80:20 for train and testing respectively. Time and date of purchase are given, so the first 80% of data is used for training and the remaining 20% for testing.

- **Hybrid Model**

For the H & M fashion dataset results using LightFM are shown in Table 7.5.

Table 7.5: Result of LightFm on H & M fashion dataset

Learning	Loss Func	Train Time(sec)	Rec Time(sec)	AUC	Pre@K	Rec@K
adagrade	WARP	127.995	0.028	0.687	0.169	0.336
	WARP_KOS	137.113	0.023	0.691	0.150	0.220
	BPR	131.390	0.028	0.544	0.113	0.191
adadelata	WARP	142.667	0.037	0.678	0.186	0.310
	WARP_KOS	150.533	0.024	0.677	0.139	0.148
	BPR	150.680	0.026	0.677	0.142	0.268

- **GMF**

For the H & M fashion dataset results using GMF are shown in Table 7.6.

Table 7.6: Result of GMF on H & M fashion dataset

Iter		Train Time(sec)	AUC	Pre@K	Rec@K	Accuracy
10	Normal weights	202	0.50	0.117	0.130	0.126
10	Non-neg weights	263	0.51	0.117	0.131	0.191
20	Normal weights	317	0.53	0.117	0.116	0.194
20	Non-neg weights	465	0.52	0.117	0.132	0.198

- **NCF**

For the H & M fashion dataset results using NCF are shown in Table 7.7.

Table 7.7: Result of NCF on H & M fashion dataset

Iter	Train Time(sec)	AUC	Pre@K	Rec@K	Accuracy
10	323	0.53	0.127	0.142	0.193
20	623	0.543	0.126	0.129	0.117

- **NeuMF**

For the H & M fashion dataset results using NeuMF are shown in Table 7.8.

NeuMF is a collaborative filtering algorithm that typically works with explicit rating data, but using hybrid matrix factorization it can work on article data. However, the H & M dataset only contains transaction data and the article dataset,

Table 7.8: Result of NueMF on H & M fashion dataset

Iter	FC layer	Train Time(sec)	AUC	Pre@K	Rec@K	Accuracy
10	1	517	0.62	0.16	0.23	0.16
20	1	803	0.65	0.16	0.24	0.11
10	3	546	0.68	0.17	0.29	0.11
20	3	1045	0.69	0.18	0.31	0.11

which means NeuMF is trained on the transaction and article dataset giving better results than collaborative filtering because it uses matrix factorization to recommend.

- **DeepWalk**

Collaborative and content-based recommendations can be improved by using other algorithms. One of the techniques that can be used is graph-based recommendation methods. To use graph-based algorithm, first step is to transform the transaction dataset into graph data. For this, the buy-together approach is used, which involves grouping items across transactions for a user as buy-together items. Once the transaction dataset is successfully transformed into buy-together items data and it can be represented as a graph where nodes are items and edges are frequency of buy-together. Then the DeepWalk algorithm is implemented to analyze the H & M dataset. The DeepWalk algorithm is a graph embedding algorithm that can extract latent features from graphs, making it well-suited for recommendation systems. This approach will produce more accurate and diverse recommendations, providing users with a more personalized and satisfying experience.

Figure 7.10: shows the buy-together data of the H&M dataset. NetworkX, a Python library known for its efficient graph representation capabilities, is used to convert the transaction dataset into graph data.

Result

For the H & M fashion dataset results using DeepWalk are shown in Table 7.9.

Table 7.9: Result of DeepWalk on H & M fashion dataset

Train Time(sec)	AUC	Pre@K	Rec@K
431	0.731	0.241	0.479

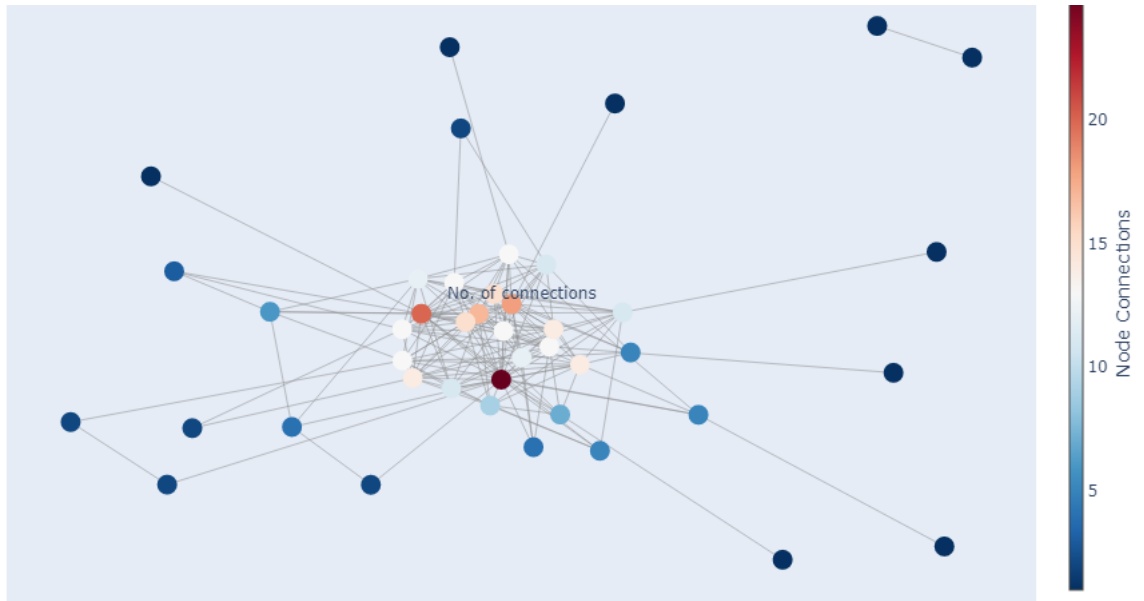


Figure 7.10: H&M fashion dataset buy-together representation using NetworkX, nodes = items and edges = frequency of buy-together

Conclusion

The H&M dataset can be characterized by a large number of users and a large number of items since the company produces and releases new clothing collections regularly. In the case of retail stores, the number of items available can vary depending on the store size and location. However, the interaction between users and items in the H&M dataset can be repeated since users may purchase an article multiple times if they like it. In such cases, collaborative filtering techniques can be used to generate personalized recommendations.

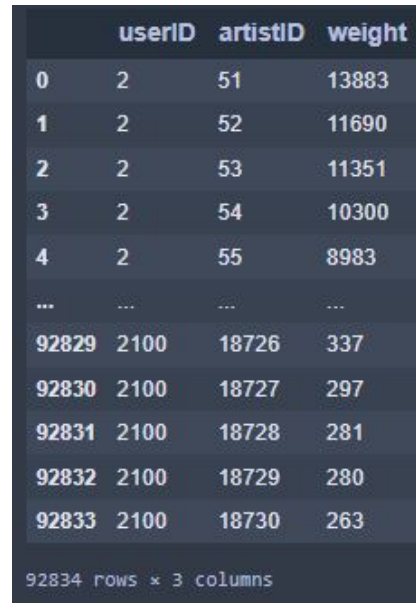
While user-user collaborative filtering can work well for recommendation systems with a rating and user data, the absence of rating data can make collaborative filtering less effective. In such cases, graph-based algorithms like DeepWalk can be used to generate recommendations based on transaction data.

Overall, the recommendation algorithms that work best for the H&M dataset may depend on the specific characteristics of the data available, including the presence or absence of rating data and the number of items available for recommendation.

7.3 lastfm dataset

The lastfm dataset[17] is widely used for building and evaluating recommendation systems in the music domain. It contains user listening histories and other metadata from the lastfm website, a music recommendation service. The dataset includes 92834 listening histories of 1892 lastfm users, covering approximately 17,000 artists.

Exploratory Data Analysis



	userID	artistID	weight
0	2	51	13883
1	2	52	11690
2	2	53	11351
3	2	54	10300
4	2	55	8983
...
92829	2100	18726	337
92830	2100	18727	297
92831	2100	18728	281
92832	2100	18729	280
92833	2100	18730	263

92834 rows x 3 columns

Figure 7.11: lastfm user artist weight data

Figure 7.11: lastfm user artist weight data has 3 columns, where **User ID** is a unique identifier for the user, **Artist ID** is a unique identifier for the artist, and **Weight** is the number of times the user has listened to the artist. This file provides a record of the number number of times each user has listened to each artist. The dataset also includes artist data that provides additional metadata about the artists, such as their names, tags, and other descriptive information.

There are **1892** users and **18632** artists. There are **92834** user-artist relations. On an average each user listens to **49** artists. On average, each artist is listened to by **5.2** users.

Figure 7.12 shows the histogram of the number of users who listen to a certain number of artists.

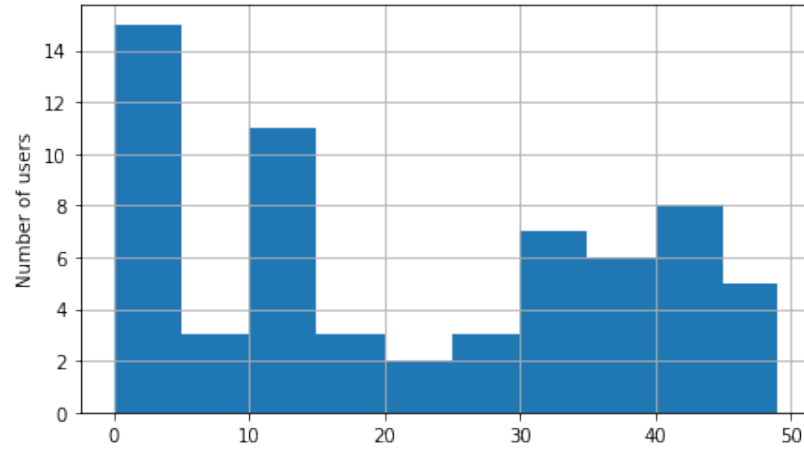


Figure 7.12: Histogram of number of users who listen to a certain number of artists

Figure 7.13 shows the histogram of the number of artists listened to by a certain number of users.

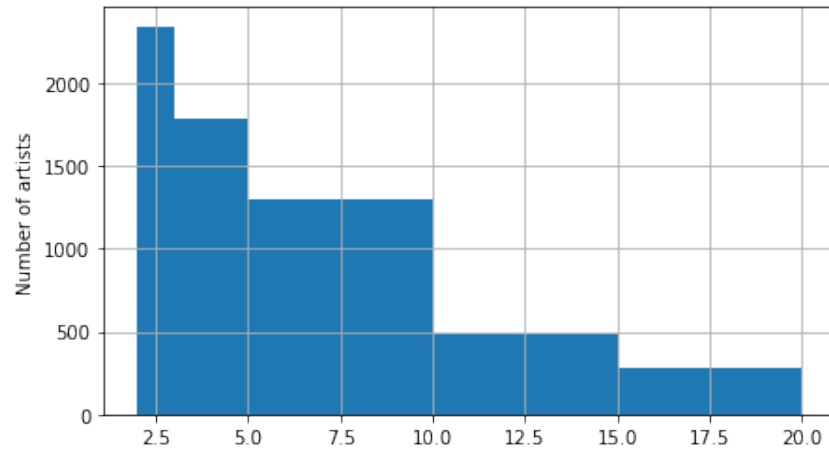


Figure 7.13: Histogram of number of artists listened to by a certain number of users

Implementation and results of algorithms

The dataset is split 80:20 for train and testing respectively. The dataset is divided randomly in such a way that shape of the interaction matrix is the same for both training and testing.

- **Hybrid Model**

For the lastfm dataset results using LightFM are shown in Table 7.10.

Table 7.10: Result of LightFM on lastfm dataset

Learning	Loss Func	Train Time(sec)	Rec Time(sec)	AUC	Pre@K	Rec@K
adagrad	WARP	75.649	0.031	0.760	0.092	0.192
	WARP_KOS	83.100	0.025	0.794	0.071	0.173
	BPR	75.542	0.028	0.706	0.068	0.168
adadelata	WARP	72.850	0.037	0.757	0.129	0.183
	WARP_KOS	84.847	0.023	0.795	0.142	0.187
	BPR	77.805	0.029	0.770	0.179	0.188

Upon conducting an analysis of the LightFM recommendation system on lastfm, it has become clear that the choice of learning model substantially impacts performance. Specifically, it has been observed that Adadelata generally gives superior Precision@K for specific applications. the WARP_KOS loss function is particularly well-suited to handling the unique characteristics of the lastfm dataset.

- **GMF** For the lastfm dataset results using GMF are shown in Table 7.11.

Table 7.11: Result of GMF on lastfm dataset

Iter		Train Time(sec)	AUC	Pre@K	Rec@K	RMSE
10	Normal weights	202	0.670	0.171	0.253	0.513
10	Non-neg weights	198	0.633	0.172	0.265	0.516
20	Normal weights	312	0.697	0.174	0.267	0.453
20	Non-neg weights	299	0.651	0.179	0.269	0.427

- **NCF**

For the lastfm dataset results using NCF are shown in Table 7.12.

Table 7.12: Result of NCF on lastfm dataset

Iter	Train Time(sec)	AUC	Pre@K	Rec@K	RMSE
10	204	0.73	0.17	0.24	0.19
20	329	0.74	0.17	0.22	0.11

- **NeuMF**

For the lastfm dataset results using NeuMF are shown in Table 7.13.

Table 7.13: Result of NeuMF on lastfm dataset

Iter	FC layer	Train Time(sec)	AUC	Pre@K	Rec@K	RMSE
10	1	309	0.741	0.170	0.260	0.180
20	1	519	0.752	0.172	0.263	0.176
10	3	495	0.733	0.175	0.283	0.210
20	3	798	0.753	0.176	0.284	0.181

NeuMF has been shown to work well on the lastfm dataset. Adding extra layers to the NeuMF architecture can improve its performance on the lastfm dataset, improving the AUC performance on the lastfm dataset.

- **ALS**

For the lastfm dataset results using NeuMF are shown in Table 7.14.

Table 7.14: Result of ALS on lastfm dataset

Train Time(sec)	AUC	Pre@K	Rec@K	RMSE
254	0.83	0.172	0.271	0.192

Conclusion

The lastfm dataset is characterized by a small number of users, a large number of items (artists), and repeated interaction because users can listen to an artist multiple times. In such cases, a content-based recommendation system can be effective since the user-item interaction data is available. Content-based recommendation systems recommend items based on the user’s past preferences or behavior, and they work well when the dataset has a large number of items and the user’s preferences are well-defined. Therefore, in the case of lastfm, where the dataset has a large number of artists, a content-based system can provide more relevant recommendations.

Matrix factorization algorithms like ALS (Alternating Least Squares) can also work well on such datasets. ALS is an algorithm used in recommendation systems that can handle missing data and provide accurate recommendations.

Overall, both content-based systems and matrix factorization algorithms can work well on the lastfm dataset, given its specific characteristics.

7.4 Dunnhumby dataset

The Dunnhumby[18] data set has been used for studying customer behavior in a superstore market. This study uses the dataset to develop and evaluate recommendation systems in the retail domain. The dataset provides transaction data that can be leveraged to build recommendation models. Specifically, the dataset includes information on customers' purchase history, which can be used to develop customer profiles and identify patterns in purchasing behavior.

Exploratory Data Analysis

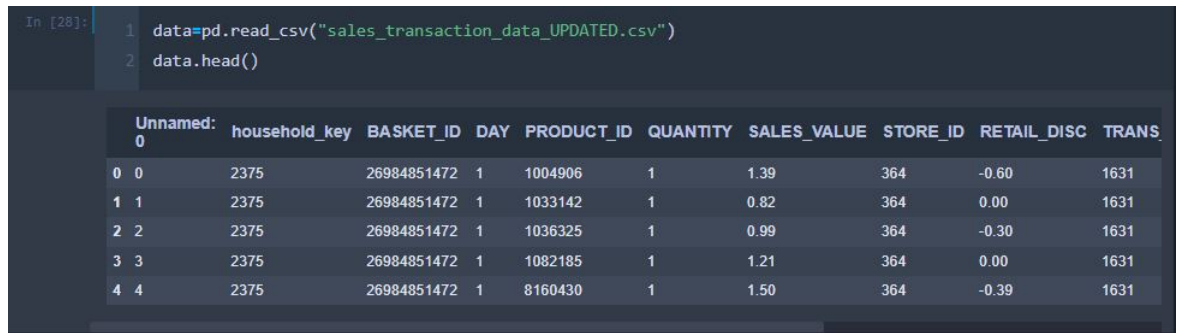


Figure 7.14: Dunnhumby Transaction dataset

A **basket** is defined as items purchased together in a single transaction, by a customer. There are **658219** transactions, as shown in Figure 7.14. There are **2456** customers, who purchased **99927** baskets, containing **4446** unique products. There are **28608** baskets that have only one product and **58** costumers who buy only one Basket

Figure 7.15 shows the histogram of the number of products in a particular basket.

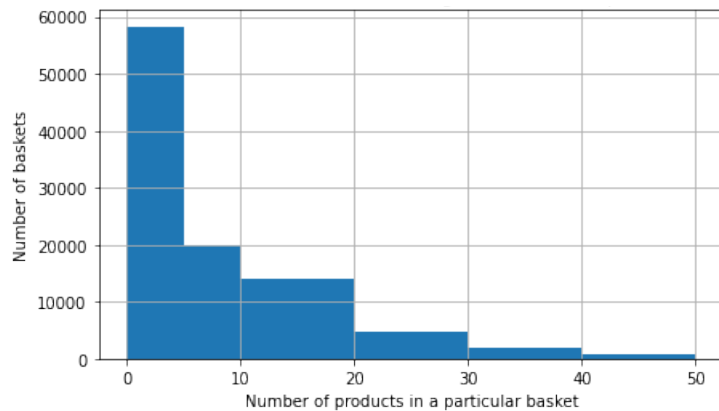


Figure 7.15: Histogram of number of products in a particular basket

Figure 7.16 shows the histogram of the number of baskets bought by a given customer.

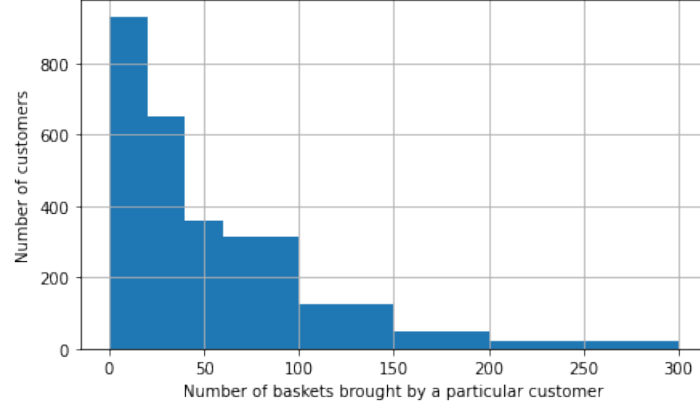


Figure 7.16: Histogram of number of baskets bought by a given customer

The Dunnhumby dataset is a transactional data source that can be transformed into a comprehensive map of product associations. Analyzing the transactional data makes it possible to determine which products are commonly purchased together and utilize this information to construct a graph of interconnected items. This graph features nodes representing individual products and edges indicating the frequency at which they were purchased together. The resulting visualization provides a representation of consumer behavior and purchasing trends.

Figure 7.17 is constructed using the python library NetworkX, which shows the graph of buy-together data where nodes are items and edges are frequency of buy-together.

By converting transactional data into a graph using NetworkX, one can leverage algorithms like DeepWalk to unlock the full potential of the data. By implementing DeepWalk on this transformed dataset, one can generate item embeddings that enable the recommendation of similar products to customers.

The dataset is split is 80:20 for train and testing respectively. Time and date of purchase are given, so the first 80% of data is used for training and the remaining 20% for testing.

- **DeepWalk**

For the Dunnhumby dataset, results using DeepWalk are shown in Table 7.15.

Employing the technique of Deep Walk, an impressive result was obtained, as an AUC score of 83% was achieved, which means that 83% of the recommended products were identified as identical to the purchased items. Additionally, the precision level of 41% was attained, signifying that out of 100 recommended items, 41 products were accurately predicted to be purchased by the user.

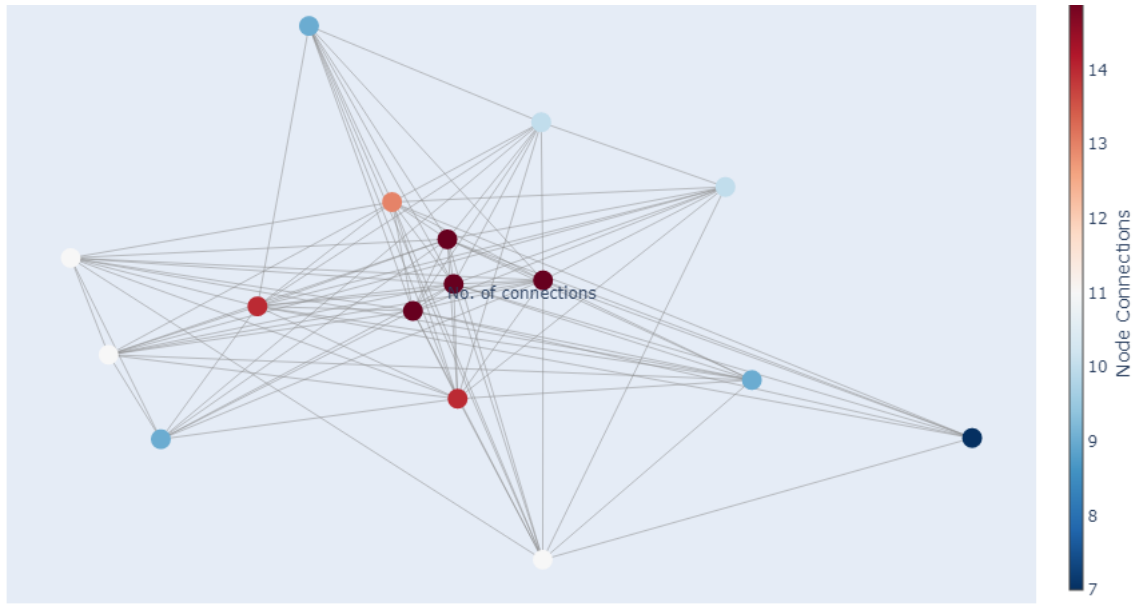


Figure 7.17: Dunnhumby dataset buy-together representation using NetworkX, nodes = items and edges = frequency of buy-together

Table 7.15: Result of DeepWalk on Dunnhumby dataset

Train Time(sec)	AUC	Pre@K	Rec@K
431	0.831	0.417	0.579

Conclusion

The Dunnhumby dataset is characterized by a large number of users, a small number of items, and repeated interaction. The number of items available in stores is limited due to space constraints, while the number of users can be many. However, a user may repeatedly purchase a product. In such cases, user-user collaborative filtering techniques can be used to generate personalized recommendations. Furthermore, graph-based algorithms like DeepWalk can be used to generate recommendations based on the transaction data available in the Dunnhumby dataset. These algorithms can leverage the relationships between products and customers to generate more accurate and relevant recommendations.

Friend recommendations in Social Network

In social networking, the interactions of individuals are interpreted through the use of graphs. These graphs are composed of nodes, each corresponding to a particular person, and edges representing the connections between them.



Figure 8.1: Friend Recommendations in Social Network [Ref [19]]

Link prediction involves predicting new edges or connections that may form in the future between individuals who are already present in the network. This is typically done using machine learning algorithms that analyze the patterns in the existing network

and identify potential connections that are likely to form.

Friend recommendation is one example of link prediction in social networks, where the goal is to recommend new friends to a user based on their existing connections and interests. This is often done by analyzing the user's activity and interests, as well as the activity and interests of their existing friends, and identifying potential new connections that share similar interests or characteristics.

Social network graphs provide a powerful framework for analyzing and understanding the complex interactions between individuals in a social network, and link prediction is just one of the many applications of this approach.

8.1 Facebook Friend Dataset

This is the Dataset[20] of Facebook social circles. Facebook data was collected from survey participants using Facebook App. The Dataset includes node features (users), circles, and ego networks.

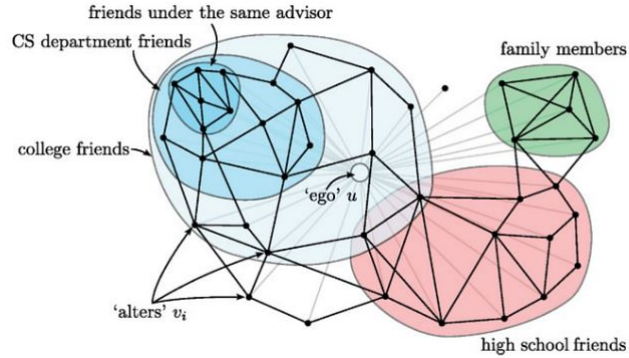


Figure 8.2: Facebook social circles [Ref[21]]

As illustrated in Figure 8.2, a graphic illustration presents a complex social network. At the center of the network is an esteemed ego user, referred to as **user u**, who is seamlessly connected to all other users in the graph, resulting in an intricate and interwoven tapestry of social connections.

Interestingly, this social network is further enriched by a diverse array of circles, each defined by a shared set of interests, attributes, and bonds between its members. These circles include high school friends, college friends, department friends, family members,

and many more, each providing a unique and fascinating glimpse into this network's social connections.

Notably, these circles may overlap, further highlighting this social network's intricate and complex nature.

8.2 Using Graph Properties and Classification algorithms

Data can be represented as a graph where nodes represent people and edges depict their relations. By utilizing the graph's properties of nodes and edges, one can gain insights into the underlying relationships and patterns within the data.

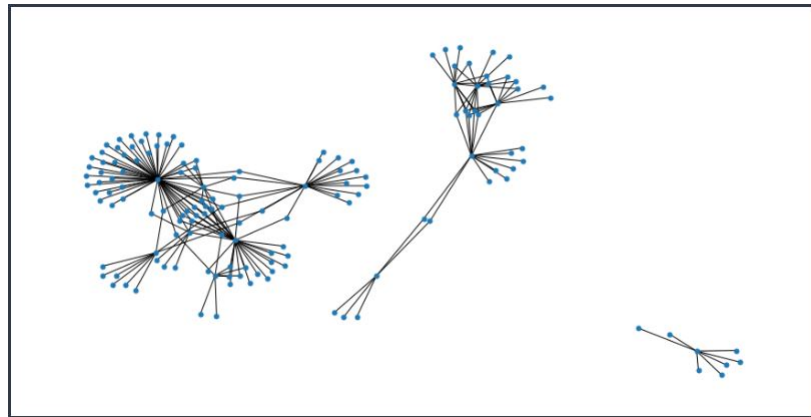


Figure 8.3: Graph Representation of Facebook Data using NetworkX

In Figure 8.3, it can be seen that there are many groups in the social network of Facebook friends. By representing the data as a graph, extracting features for each link becomes possible, and these features can be used for link prediction between users. Some links indicate a connection, while others are negative, meaning there is no connection. Using these features, a classification model can be implemented to predict user links effectively.

The following steps were taken:

Step 1: Negative edges were added to represent the absence of connection between users, thus enhancing the dataset with a more realistic scenario.

Step 2: Features were extracted for both positive and negative links, enabling the model to distinguish between the two classes of edges.

Step 3: A classification model was built using the properties of the edges, enabling the model to classify the edges as positive or negative with high accuracy.

Step 4: For edge prediction, the model first extracts all features of the given edge and then employs the classification model to determine whether the edge belongs to the positive or negative class. If the edge is positive, it recommends user u to user v . Otherwise; it does not make any recommendations. This approach ensures that the recommended connections are highly relevant and are likely to result in positive outcomes for the users involved.

The properties utilized in link prediction are as follows. [22]

- **Common neighbors**

In graph theory, the common neighbor[22] is a concept that refers to the number of shared neighbors between two nodes in a graph. Given two nodes, A and B, on a graph, the common neighbors of A and B are the nodes connected to both A and B. The number of common neighbors is the size of this set.

- **Resource allocation index**

The Resource Allocation Index (RAI)[22] is used in network analysis to evaluate the potential of nodes in a graph to transfer resources to each other. It is based on the principle that nodes with more connections can better share resources with other nodes in the network. The RAI is calculated by summing the contributions of all neighbors of a given node, where the contribution of each neighbor is proportional to its degree in the graph.

The formula to calculate the Resource Allocation Index (RAI) in a graph is as follows:

$$RAI(A, B) = \sum_{k \in N(A) \cap N(B)} \frac{1}{deg(k)}$$

where:

- $RAI(A, B)$ is the RAI between nodes A and B.
- $N(A)$ represents the set of neighbors of node A.
- $N(B)$ represents the set of neighbors of node B.
- $deg(k)$ represents the degree of node k.

- **Jaccard coefficient**

The Jaccard coefficient[22] is a commonly used similarity measure in graph theory, used to compare the similarity between two sets of nodes in a graph. In graph

theory, the Jaccard coefficient is the ratio of common neighbors between two nodes to the total number of neighbors those two nodes have.

For example, consider two nodes, A and B, in a graph, and let $N(A)$ and $N(B)$ denote the set of neighbors of A and B, respectively. The Jaccard coefficient between A and B is defined as

$$J(A, B) = |N(A) \cap N(B)| / |N(A) \cup N(B)|$$

where $|S|$ denotes the cardinality of set S.

- **Adamic adar index**

The Adamic-Adar index[22] is a measure of the similarity between two nodes in a graph based on the common neighbors of those nodes. The formula to calculate the Adamic-Adar index for nodes u and v is as follows:

$$AAI(A, B) = \sum_{k \in N(A) \cap N(B)} \frac{1}{\log(\deg(k))}$$

Here, the summation is taken over all common neighbors k of A and B, and $\deg(k)$ is the degree of node k (i.e. the number of edges incident to it).

- **Preferential attachment**

Preferential attachment[22] is a mechanism commonly used to model network growth, where the probability that a new node attaches to an existing node is proportional to its degree (i.e., the number of edges it has). The preferential attachment formula for node i in a graph G can be expressed as:

$$P(i) = \frac{k_i}{\sum_j(k_j)}$$

Where $P(i)$ is the probability of a new node attaching to node i, k_i is the degree of node i (i.e., the number of edges it has), and $\sum_j(k_j)$ is the sum of degrees of all nodes in the graph G.

- **Common neighbor centrality**

Common Neighbor Centrality[23] is a measure used in network analysis to quantify the extent to which two nodes in a graph share common neighbors. It is calculated using the following formula:

$$CNC(u, v) = \alpha |N(u) \cap N(v)| + (1 - \alpha) \frac{N}{d_{uv}}$$

Where $CNC(u,v)$ represents the standard neighbor centrality between nodes u and v , α is the parameter varies between $[0,1]$, $N(u)$ and $N(v)$ denote the set of neighbors of nodes u and v , respectively, and d_{uv} denotes the shortest distance between u and v .

	CN	RAI	JC	AAI	PA	CNC	label
0	0	0.596562	0.466667	14.558623	12036	459.9	1
1	1	0.348868	0.302326	5.948752	3120	424.7	1
2	2	0.130195	0.166667	1.544083	272	408.7	1
3	3	0.755404	0.644670	24.706558	26144	505.5	1
4	2	0.387228	0.290000	6.629850	3920	427.1	1
...
2110	2	0.168244	0.333333	1.589586	143	408.7	0
2111	3	0.241155	0.352941	2.972908	504	413.5	0
2112	2	0.427043	0.583333	2.347492	90	409.5	0
2113	0	0.320411	0.337500	6.034641	2820	425.5	0
2114	6	1.058092	0.584071	15.714132	7828	456.7	0

Figure 8.4: Extracted features using graph properties

In Figure 8.4: **CN** means common neighbors, **RAI** means resource allocation index, **JC** means Jaccard coefficient, **AAI** means Adamic adar index, **PA** means preferential attachment, **CNC** means common neighbor centrality

Following models are used for classification.

- **SVM**

The support vector machine (SVM)[24] is a supervised machine learning algorithm for classification. It employs the technique of identifying a hyperplane in an N-dimensional space that effectively separates and distinguishes the data points. When SVM is trained on the above features extracted from the graph, an accuracy of 61% and an AUC of 0.678 is obtained.

- **Logistic Regression**

Logistic Regression[25] is a supervised machine learning algorithm that is widely utilized for classification purposes. This algorithm leverages multiple independent variables to predict a dependent variable with impeccable accuracy. Its potential lies in the ability to scrutinize the intricate relationship between the independent and dependent variables, allowing it to make astute predictions based on the

patterns observed. When logistic regression is trained on the above features extracted from the graph, an accuracy of 67% and an AUC of 0.645 is obtained.

- **Artificial Neural Network (ANN)**

The Artificial Neural Network[26] is a machine learning technique that excels at classification and regression problems. This technique is characterized by interconnected input-output networks, where each connection is assigned a weight. These networks comprise one input layer, one or more intermediate layers, also known as hidden layers, and one output layer. The complexity of this network allows it to learn complex patterns and relationships in data, making it an essential tool for data scientists and machine learning practitioners.

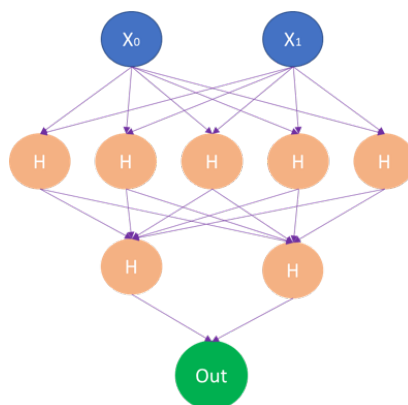


Figure 8.5: Artificial Neural Network [Ref[27]]

Figure 8.5 shows an artificial neural network. In the image, blue is the input layer, orange denotes hidden layers and green is the output layer. When trained on FB friends' data, the ANN has an accuracy of 69% and an AUC of 0.672.

8.3 Graph Neural Network (GNN)

A Graph Neural Network (GNN)[21] is a neural network operating on graph-structured data. It is handy when dealing with data characterized by the relationships between entities, such as social networks, molecule structures and recommendation systems.

One defining feature of GNNs is how they handle both node and edge data. In traditional neural networks, the input is usually a fixed-length vector. In contrast, GNNs can take input graphs with varying numbers of nodes and edges, where each node and edge can have its data associated with it.

GNNs typically operate by iteratively updating the features of each node in the graph based on the characteristics of its neighbors. This process can be repeated multiple times to allow the network to incorporate information from increasingly distant parts of the graph.

Using node and edge data in GNNs makes them particularly useful for node classification, link prediction, and graph classification. They have been successfully applied in various domains, including computer vision, natural language processing and drug discovery.

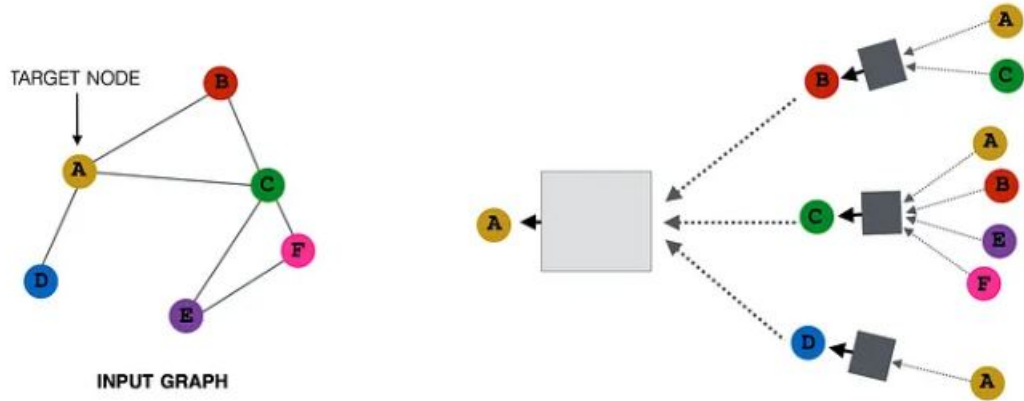


Figure 8.6: Graph Neural Network[Ref[21]]

Graph Neural Network (GNN) model computes and aggregates messages from neighboring nodes in a graph. For example, in Figure 8.6, Node A is the target node, with Nodes B, C, and D as its neighbors. The neighbors of B, C, and D are (A, C), (A, B, E, F) and (A), respectively.

The GNN model begins by computing the message from Nodes A and C and then aggregates the messages. The aggregated information is then passed to Node B, while Nodes C and D get updated through message-passing. The model then computes the messages from Nodes B, C, and D and aggregates the messages. Finally, the aggregated information is passed back to Node A. This special two-layer message passing the GNN approach is repeated for all nodes in the graph.

The message-passing operation referred to here exhibits permutation equivariance, meaning that the computation graph for the target node remains constant for differ-

ent order plans. This remarkable property allows each node to gather and accumulate information from its local neighborhood. Utilizing models such as GCN and GraphSAGE, generating node embeddings for every single node within the network becomes possible. These embeddings can then be leveraged for link prediction, enabling further insights and analysis of the network’s behavior.

8.3.1 Graph Convolutional Neural Network (GCN)

The Graph Convolutional Neural Network (GCN)[21] is a type of neural network that operates on graph data, allowing for the analysis of complex network structures. By generalizing convolution operation from the grid to graph data, GCN is a tool for analyzing complex network structures, providing a framework for modeling and predicting a wide range of phenomena.

GCN is built on graph convolutions, constructed by stacking multiple convolutional layers, each followed by a pointwise nonlinearity function. This layer-by-layer approach allows for the efficient analysis of complex network structures, enabling researchers to uncover new insights into the underlying properties and dynamics of graph data.

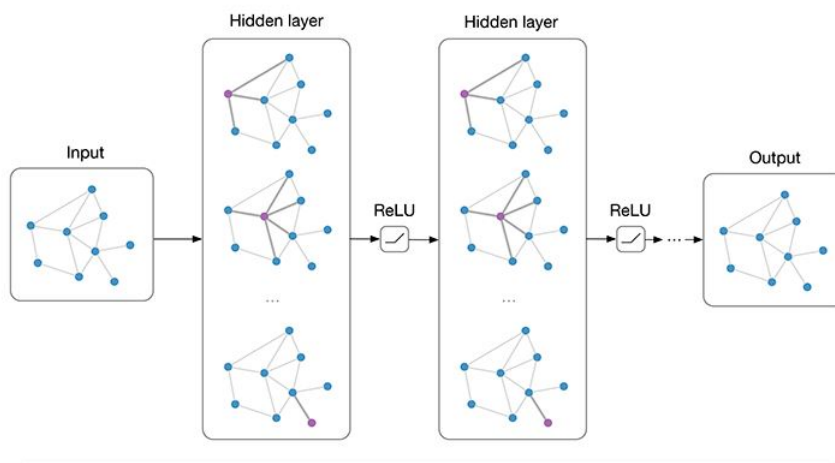


Figure 8.7: Graph Convolutional Neural Network [Ref[28]]

8.3.2 GraphSAGE (Graph Sample and aggreGatE)

GraphSAGE[21] is a graph neural network model that learns to generate node embeddings by aggregating and combining information from a node’s local neighborhood. The

model leverages an efficient sampling scheme to understand scalable representations for large graphs.

The GraphSAGE model first samples a fixed number of neighbor nodes for each node in the graph and then generates embeddings by aggregating the embeddings of the sampled nodes. The aggregation function is learned using a graph convolutional neural network, allowing the model to capture higher-order structural information about the graph.

GraphSAGE has been used for node classification, link prediction, and graph classification. Its ability to learn scalable representations for large graphs makes it a valuable tool for analyzing complex network structures and modeling a wide range of phenomena.

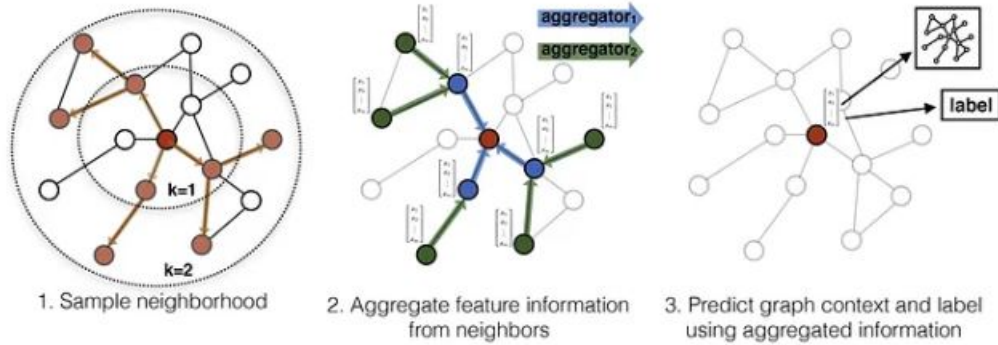


Figure 8.8: GraphSAGE [Ref[29]]

8.3.3 Results

After obtaining node embeddings through the GNN model, the next crucial step towards friend suggestion is to model the connection between nodes as a link prediction task. This involves estimating a likelihood score for each potential edge in the graph, determining whether a friendship exists between two users.

One way to obtain this score is by taking the dot product between the learned embeddings of the two nodes. Although not an accurate estimate of probability, this approach has proven effective in predicting the existence of edges in the graph. For GraphSAGE, AUC is 0.897 and for GCN, AUC is 0.931

8.4 Results

Table 8.1 summarizes the results from different models implemented on the Facebook friends dataset for link prediction.

Accuracy means predicting an edge correctly between two nodes.

	AUC	Accuracy
SVM	0.678	0.61
Logistic Regression	0.645	0.67
ANN	0.672	0.69
GCN	0.931	0.80
GraphSAGE	0.897	0.78

Table 8.1: Friend Recommendations in Social Network

8.5 Conclusion :

It can be asserted that GNN models, particularly Graph Convolutional Neural Networks (GCN) are more accurate in processing Facebook data with social networks than classification models. GNN models leverage node features and relationship data, allowing for a more comprehensive analysis of complex network structures. As a result, GNN models have been found to outperform classification models in terms of accuracy. In particular, using GCN models has significantly improved accuracy for link prediction tasks.

Conclusions

In conclusion, this project has highlighted the importance of selecting the appropriate recommendation algorithms based on the characteristics of the dataset and the type of problem at hand. Collaborative filtering is effective for datasets with a large number of users, a small number of items, and one-time interaction, especially if rating data is available. Hybrid recommendation systems are also useful when item data is available in addition to rating data.

For datasets with a small number of users, a large number of items, and one-time interaction, content-based recommendation systems are the way to go. On the other hand, matrix factorization recommendation systems are more effective when dealing with a large number of users, a large number of items, and one-time interactions.

Graph-based algorithms like DeepWalk are useful in situations where rating data is unavailable, but purchase or transaction data is available. Finally, for friend recommendation problems on social networks like Facebook, Graph Convolutional Neural Networks (GCN) and GraphSAGE are superior to classification models due to their ability to leverage node features and relationship data in complex network structures.

In summary, selecting the appropriate recommendation algorithm is crucial for accurate and relevant recommendations, and this project has provided insights into which algorithms work best for different types of datasets and problems.

Future work

Based on the analysis of the different recommendation systems and the algorithms used for them, future work can focus on developing more advanced algorithms that can

handle complex data sets with multiple types of interaction and data.

One area for improvement is to explore more advanced graph-based algorithms, such as Graph Neural Networks (GNNs) that can handle heterogeneous graphs with multiple types of nodes and edges. These algorithms can be used to generate more accurate and relevant recommendations in scenarios where the data is complex and diverse, such as in social networks, e-commerce platforms, and online communities.

List of Figures

2.1	Screenshot from the Movie Lens [Ref[1]]	11
2.2	Screenshot from the Amazon [Ref[2]]	12
3.1	Types of Recommendation Systems	15
3.2	Content-based Recommendation System [Ref[4]]	17
3.3	User-based Collaborative Filtering [Ref [4]]	18
3.4	Item-based Collaborative Filtering [Ref[4]]	19
3.5	Matrix factorization [Ref[4]]	20
3.6	Hybrid Recommendation System [Ref [5]]	21
6.1	Architecture of GMF	33
6.2	Neural Collaborative Filtering [Ref [12]]	34
6.3	NeuMF Architecture [Ref [12]]	35
7.1	Movie-Lens rating dataset	40
7.2	Movie-Lens movies dataset	40
7.3	Histogram of number of users who watched a particular movie	41
7.4	Histogram of number of movies watched by a particular user	41

7.5	Number of movies for each genre	42
7.6	H & M fashion article data set	45
7.7	H & M fashion transaction data set	45
7.8	Histogram of number of articles purchased by a particular customer . .	46
7.9	Histogram of the number of customers who bought a particular article .	46
7.10	H&M fashion dataset buy-together representation using NetworkX, nodes = items and edges = frequency of buy-together	49
7.11	lastfm user artist weight data	50
7.12	Histogram of number of users who listen to a certain number of artists	51
7.13	Histogram of number of artists listened to by a certain number of users	51
7.14	Dunnhumby Transaction dataset	54
7.15	Histogram of number of products in a particular basket	54
7.16	Histogram of number of baskets bought by a given customer	55
7.17	Dunnhumby dataset buy-together representation using NetworkX, nodes = items and edges = frequency of buy-together	56
8.1	Friend Recommendations in Social Network [Ref [19]]	57
8.2	Facebook social circles [Ref[21]]	58
8.3	Graph Representation of Facebook Data using NetworkX	59
8.4	Extracted features using graph properties	62
8.5	Artificial Neural Network [Ref[27]]	63
8.6	Graph Neural Network[Ref[21]]	64
8.7	Graph Convolutional Neural Network [Ref[28]]	65
8.8	GraphSAGE [Ref[29]]	66

List of Tables

7.1	Result of LightFm on movie lens dataset	42
7.2	Result of GMF on movie lens dataset	43
7.3	Result of NCF on movie lens dataset	43
7.4	Result of NueMF on movie lens dataset	43
7.5	Result of LightFm on H & M fashion dataset	47
7.6	Result of GMF on H & M fashion dataset	47
7.7	Result of NCF on H & M fashion dataset	47
7.8	Result of NueMF on H & M fashion dataset	48
7.9	Result of DeepWalk on H & M fashion dataset	48
7.10	Result of LightFM on lastfm dataset	52
7.11	Result of GMF on lastfm dataset	52
7.12	Result of NCF on lastfm dataset	52
7.13	Result of NeuMF on lastfm dataset	53
7.14	Result of ALS on lastfm dataset	53
7.15	Result of DeepWalk on Dunnhumby dataset	56
8.1	Friend Recommendations in Social Network	67

Bibliography

- [1] “Movie lens website,.” [URL](#).
- [2] “Amazon book website.” [URL](#).
- [3] D. Das, L. Sahoo, and S. Datta, “A survey on recommendation system,” *International Journal of Computer Applications*, vol. 160, no. 7, 2017.
- [4] “Content-based recommendation system image.” [URL](#).
- [5] “A guide to building hybrid recommendation systems for beginners.” [URL](#).
- [6] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, “How good your recommender system is? a survey on evaluations in recommendation,” *International Journal of Machine Learning and Cybernetics*, vol. 10, pp. 813–831, 2019.
- [7] “Evaluating recommender systems: Root means squared error or mean absolute error?.” [URL](#).
- [8] “Recall and precision at k for recommender systems.” [URL](#).
- [9] “Common metrics to evaluate recommendation systems.” [URL](#).
- [10] “Matrix factorization for recommender systems.” [URL](#).
- [11] “Lightfm documentation.” [URL](#).
- [12] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, “Neural collaborative filtering,” in *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.
- [13] “A gentle introduction to alternating least squares.” [URL](#).

- [14] B. Perozzi, R. Al-Rfou, and S. Skiena, “Deepwalk: Online learning of social representations,” in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.
- [15] “Movie lens dataset.” [URL](#).
- [16] “H and m fashion dataset.” [URL](#).
- [17] “Last.fm dataset.” [URL](#).
- [18] “Dunnhumby dataset.” [URL](#).
- [19] “MI behind facebook’s friend suggestions.” [URL](#).
- [20] “Social circles: Facebook.” [URL](#).
- [21] “Friend recommendation using graphsage.” [URL](#).
- [22] “Link prediction.” [URL](#).
- [23] “Common neighbor centrality.” [URL](#).
- [24] “Support vector machines.” [URL](#).
- [25] “Logistic regression — detailed overview.” [URL](#).
- [26] “Artificial neural network in tensorflow.” [URL](#).
- [27] “Nn - multi-layer perceptron classifier (mlpclassifier).” [URL](#).
- [28] “Graph convolutional networks.” [URL](#).
- [29] “Graph neural networks: A deep neural network for graphs.” [URL](#).