Data Analytics, MSc Dissertation MTHM038, 2022/23

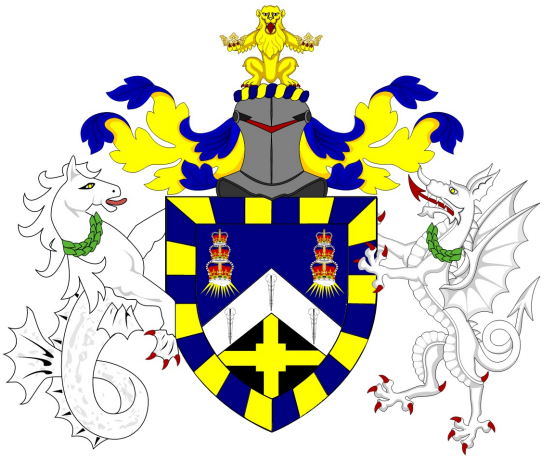
Collaborative Filtering Recommendation Systems

A comparative study on User and Item based models

Adrika Datta

ID 220997078

Supervisor: Dr. Argyro Mainou



A thesis presented for the degree of
Master of Science in *Data Analytics*

School of Mathematical Sciences

Queen Mary University of London

Declaration of original work

This declaration is made on September 4, 2023.

Declaration :

I , Adrika Datta hereby declare that the work in this thesis is my original work. I have not copied from any other students' work, work of mine submitted elsewhere, or from any other sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

Referenced text has been flagged by:

1. Using italic fonts, **and**
2. using quotation marks "...", **and**
3. explicitly mentioning the source in the text.

This work is dedicated to my father

Abstract

The proliferation of digital media and e-commerce has fuelled the industry's race to create the perfect recommendation system. In a highly competitive industry where retaining customers by providing useful recommendations can make the difference between overwhelming success and failure, the major players in the streaming services - Netflix, Hulu, Disney+ etc or in the e-commerce space where brands are launching their in-house platforms rather than relying on tech giants like Amazon or eBay, are fighting for market share. One of the key aspects of their business models are to provide precise recommendations to users/customers of products that they are likely to interact with or purchase thus driving revenues and share prices. Market domination for these tech companies has never been harder. For the customer, they are much more likely to stick to platforms that cater to their unique preferences, and has the ability to switch in matter of seconds. Therefore companies invest heavily in these systems. The "recommendations" often are an amalgamation of the user's previous purchase history, feedback from users who fall in the same "cluster" or feature of items in their database. Although significant advances in these algorithms are being done, it is still an active area of research where the aim is to solve the several drawbacks it comes with - cold start, data sparsity, heavy computational costs arising from scalability to mention a few. This work takes a deep dive into the implementation and performance of user-based & item-based Collaborative Filtering(CF) systems. These algorithms are tested on two publicly available datasets - Movie Lens and Book Recommendation Data sourced from Kaggle. The key findings of this thesis are the superiority of item-based CF models over user-based models, the significance of implicit feedback in the dataset and using a fraction of the dataset does not lead to significant performance loss during inference time.

Acknowledgements

The Master's thesis is a journey through several highs and lows. I'm grateful for the wonderful individuals I had by my side throughout this journey. Firstly, I'd like to thank my thesis advisor, Dr. Argyro Mainou, the freedom you gave me to pursuit my interests and follow my curiosity made all the difference. Thank you for your support through thick and thin even under virtual working conditions.

I am grateful for the support from my family and friends at home in Kolkata, India and here in the United Kingdom. Special thanks to my parents Mr. Anup Kumar Datta and Late Shampa Datta for everything that I have today. I'd also like to thank my partner, Sammya, who stood by me through my darkest moments and finest hours.

I am very fortunate to have met mentors along the way who helped me grow. Special thanks goes to Dr. Nicola Perra and Dr. Nina Otter – the classes that I took taught by them enabled me to cultivate valuable research skills. I'd like to express my gratitude to Andrew Ng, for his educational endeavors where I learned how machine learning works and is applied in the real world.

I am grateful for all the amazing people that I got the chance to know. I would like to thank Martin Benning for his feedback on drafts of this thesis. Lastly, I would like to express my gratitude to the Department of Mathematical Sciences at Queen Mary University of London for a wonderful journey of academic pursuit.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

In 1992, Belkin and Croft examined and analyzed information filtering and information retrieval. According to [2] information retrieval is the fundamental technology of search engine, and the recommendation systems are mainly based on the technology of information filtering. In the same year, Goldberg et al. presented the [6] Tapestry system which is the first information filtering system based on collaborative filtering through human evaluation. In 1998, Giles et al. published a paper on recommender systems as part of the CiteSeer project [3], the foundation of the contemporary recommendation systems traces its roots back then, wherein their initial implementation revolved around experimental utilization in the realms of personal email and information filtering.

In the present era, three decades hence, personalized recommendations have become an omnipresent phenomenon, permeating numerous domains, and garnering unprecedented success. Consequently, the field of artificial intelligence, particularly concerning this thriving application domain, is witnessing an exponential surge in research activities.

Recommendation systems have emerged as vital tools in the digital landscape, catering to the need of predicting user preferences and enriching their online experiences amidst a vast array of choices. Whether they explicitly offer tailored recommendations, as exemplified by prominent platforms like Amazon, Spotify and Netflix, or operate subtly behind the scenes to curate content, their importance lies in the ability to expose users to undiscovered

options and extend their engagement.

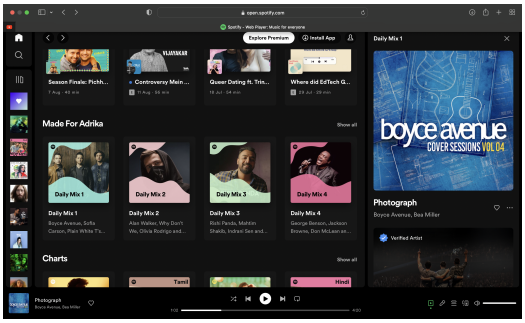


Figure 1.1: Spotify Daily Mix

There are various techniques to design these systems - ranging from the simple methodologies that rely solely on user ratings of similar items to the extremely complex approaches that leverages multiple data sources. Thus, the recommendation task provides an excellent problem space to apply machine learning methodology. As users continually interact with the system and contribute additional data, these systems can evolve to further refine their recommendations, offering users personalized and highly relevant content suggestions. This thesis explores the intricate world of different types of recommendation systems, specifically focusing on the application of collaborative filtering, to enhance the efficacy and performance.

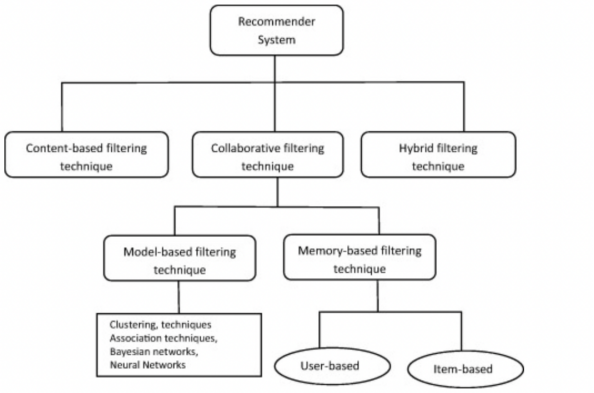


Figure 1.2: Types of Recommendation Systems [8]

1.1.1 Functions

In the business world, recommendation systems serve various crucial purposes, most notably :

- Improving User Satisfaction through *Personalization* : In the contemporary times, with the abundance of content and information available everywhere, users can easily get overwhelmed. An effective recommendation system captures user preferences in detail, either explicitly through feedback or implicitly by analyzing their behaviour to offer personalized suggestions tailored to each user's preferences, making it easier for individuals to discover relevant product, services, and content, ultimately enhancing their overall experience. (Figure : 1.1)

- *Boosting Sales* by *Cross-Selling* and *Up-Selling* : One of the main reasons company deploy recommendation systems is to increase the conversion rate. An effective recommendation system suggests the customers with additional items that can complement or enhance their main purchase based on their preferences. The focus is to encourage the users to purchase additional(result of Cross-Selling) or higher-value(result of Up-Selling) products or services thereby leading to increased revenue.

- Data-Driven *Insights* : Businesses can gain valuable insights into customer behaviour, preferences, and trends through recommendation system data. These strategies can inform marketing strategies which thereby helps in retaining customers and form a base for future products and marketing approaches.

- *Competitive Advantage* : A well-designed and effective recommendation system can contribute significantly to a company's competitive advantage by improving user experience, increasing customer retention, enabling targeted marketing strategies. and establishing a distinct brand identity, helping the business to stand out in a crowded marketplace and drive sustainable growth and success.

1.1.2 Applications

Research on recommendation systems plays a strong emphasis on real-world applications, particularly in commercial settings. The choice of algorithmic approach is heavily influenced by the specific domain in which the recommendation system is implemented. To better understand the diverse landscape of recommendation systems, we will categorize some of these existing applications into there distinct domains [11].

Table 1.1: Applications of Recommendation System [15]

Domain	**Type**	**Example**
Entertainment	Movie Music Applications	Netflix Spotify Apple Store
E-Commerce	Books Beauty supplies Clothes	Amazon Sephora Marks & Spencer
Content	Personalized Newspaper E-mail filters Image Recommendation	The Sun Outlook Pinterest
Social Media	Social Networking	Facebook Twitter Instagram
Financial Service	Investment advice, Insurance advice, Credit card offers	Revolut
Education Service	Learning Platforms	Coursera
Tourism Service	Personalized Itenaries Hotel Tour Operators	GetYourGuide Booking.com Holibob

1.1.3 Current Challenges

Generally, the most indicative measure of a recommendation system's performance is user satisfaction. Even though it is not possible to compute users' satisfaction by using an empirical formula, we can still measure it's performance based on how well they can handle common issues. In this section, we provide an understanding of the metrics used to measure the performance of a recommendation system against main challenges. Problems that commonly occur include (but are not limited to) cold-start, accuracy, data sparsity, scalability, and diversity.

1. **Data Sparsity :** Sparse data refers to a dataset where most of the possible entries are missing or empty. It is a common challenge in collaborative recommendation systems. This happens when items receive very few ratings, making it difficult for the system to suggest them to users, even if those few ratings are positive. This issue can lead to poor recommendations for users with unique tastes or for new users. To address this, demographic information like age, gender, and location can be used to find similarities among users. Associative recovery methods and spreading activation algorithms can also help find connections among users based on their past interactions [12].

2. **Cold Start Problem :** [5] The term 'cold start' comes from automobiles. When the engine is cold, it faces difficulty to start, but once it gets started and reaches an optimal temperature it runs smoothly. Similarly when there is insufficient data or metadata available, a recommendation system does not perform optimally. Cold starts can be classified into two distinct subsets: Product cold starts and User cold starts. A new product goes through a product cold start whenever it is first uploaded on an e-commerce database, due to the lack of reviews or user interaction. Similarly if there are not enough user interactions, the system would not know what to recommend. The user cold start occurs when an user creates an account for the first time and thereby does not have any product preferences or history available. The cold start problem always exists for new or existing users. For example, if John searches for a laptop on the internet and within a week, he buys one, and is no longer interested in purchasing laptops anymore; what should the recommendation system display now? Users will always be interested in new and different things.

3. **Accuracy :**

   When the movie database of a recommender system contains a limited number of movies, it tends to yield higher accuracy due to the focused information pool. Conversely, larger databases often lead to reduced accuracy as the extensive search space can dilute results. To mitigate this, the K-means algorithm optimizes computational efficiency by limiting iterations or selecting a top-N subset of movies for recommendations [19]. However, movies lacking ratings can introduce bias. Enhancing recommendation accuracy involves sophisticated algorithms that comprehensively search for matches between product features, user traits, and item characteristics.

4. **Fairness and Diversity :**

   The most accurate recommendations are made by recommending items/objects based on user or objects' similarity. This is basically overlapping instead of differences, which is the diversity issue[5]. This exposes the users to a narrower selection of objects, while highly related items may be overlooked. The diversity of recommendations allows the users to discover objects which they would not readily find for themselves. The two measures by which diversity of a recommendation system can be evaluated are : [25] - Surprisal and Personalization. 'Surprisal' measures are used to calculate the system's ability to recommend unpredictable results, which is the unexpectedness of an item proportional to its global popularity. 'Personalization' is the uniqueness of different user's recommendation lists and can be calculated by the inter-list distance. One potential concern is that if an algorithm focuses strictly on increasing diversity, accuracy decreases. Cases which are overly focused on accuracy are known as overconcentration. [5]

5. **Scalability :**

   Scalability ensures a harmonious balance between accuracy and computational efficiency. With the rapid expansion of the e-commerce sites, the recommendation systems need to generate quick results for large scale applications. For a platform that has millions of users and products, scalability is a serious issue. One-dimensionality reduction is known to be a common technique to reduce scalability issues. [22] Singular value decomposition (SVD) has also been used to reduce the scalability issue. SVD is used for dimensionality reduction.

1.2 Recommendation System

1.2.1 Problem Formulation

The utility of an item depends on the application's goal, for example, an item can be more useful if it increases user satisfaction or better capture a user's needs. In recommendation systems, an item's value is usually represented by a rating, which indicates how much a particular user liked a specific item. The aim is to infer unknown ratings from known ones. Formally, the recommendation problem can be formulated in the following manner: [20]
- Let U be the set of all users and let I be the set of all items. In practical applications, these two sets are usually very large, up to hundreds of thousands of elements.
- Let F be the utility function that measures the relevance of the item i to user u and let R be an ordered set of user preferences for the items, R = (1, 2, ... ,N), N is a typically small integer :

$$f\colon U \times I \to R$$

- For each user $u \in U$, we want to choose an item $i \in I$, that maximizes the user's utility. Mathematically, given the above notation and settings, the optimization problem is :

$$\forall u \in U, i_s = \underset{i \in I}{\operatorname{argmax}}\, F(u, i) \tag{1.1}$$

where each element of U can have various characteristics, such as user ID, gender, age, occupation etc. Similarly, item space I can have a set of characteristics, such as release year, genre, artists (in case of a music recommendation applications e.g. Spotify), or personalized itineraries, accommodation recommendations, local activities (in case of travel recommendation applications e.g. GetYourGuide). The main issue faced by the recommendation system is that the utility F is not defined over the whole $U \times I$ space - at best, we only know a limited subset of ratings because a user generally does not rate all available items, but a subset of them, typically orders of magnitude smaller than the number of available items.
- Let N_s be the set of items for which the ratings are known and $\bar{N}_s$ be it's compliment. We are interested in solving the following optimization problem:

$$\forall u \in U, i_s = \underset{i \in \bar{N}_s}{\operatorname{argmax}}\, F(u, i) \tag{1.2}$$

The recommended item, i_s, for user u, maximizes Eq.1.2 and may be different for every user. The solution of Eq.1.2 is usually obtained by first estimating the matrix R ($|U| - by - |I|$ rating matrix) on the subset $\bar{N}_s$, and then, for every user, selecting the item for which the estimated rating is the highest. In general in order to recommend $N \geq 1$ items we need to select top-N items for which the estimated ratings are the highest.

Categorically, recommendation systems slot into three primary categories :

- **Content-based Filtering :** This approach involves suggesting new items that share similar attributes with items, that the user has previously favoured. The effectiveness of these recommendations hinges on the intrinsic value of the item features.
- **Collaborative Filtering :** In this technique, the system proposes items to new users based on the preferences of existing users within the system. The accuracy of these suggestions is reliant on the reliability of user preferences data.
- **Hybrid Approach :** A hybrid system combines content-based and collaborative filtering methodologies to formulate recommendations. By using these techniques together, the system aims to enhance the overall quality of generated suggestions.

Table 1.2: Examples of different systems' [15]

Method	Name	Technique	Domain
Content-based Filtering	ACR News	URL Clusters	Netnews Filtering
Collaborative Filtering	GroupLens	User-Item Ratings Matrix	Netnews Recommender
Hybrid Approach	WebWatcher	Boolean Feature Vector	Web Recommender

Content-based Filtering

In the context of content-based recommendation systems, the utility function F(u, i) for a particular user u and item i is approximated by leveraging the utilities F(u, i_n) assigned by user u to each item $i_n \in$ I that bears resemblance to item i. For instance, in case of a book recommendation system, in order to recommend new books, a content-based recommendation approach tries to understand the similarly among the books that the user u has read frequently in the past. Then, only the books which have a high degree of similarity to the user's preferences would eventually be recommended.

Collaborative Filtering

In the context of collaborative recommendation systems, the utility function F(u, i) of item i for user u is estimated based on the utilities F(u_j, i), assigned to item i by those users $u_j \in$ U who are similar to the user u, i.e. in collaborative approach it tries to predict the utility of items for a particular user based on the items he/she has previously rated by other users. For instance, in case of a book recommendation system, in order to recommend new books to user u, the approach would be to find users similar to user u who has similar taste in books or who has similar ratings in books, then only, the books that are most liked by the similar users would be suggested to the user u.

Hybrid Approach

A hybrid system combines content-based and collaborative filtering methods, without the restrictions/limitations of each of these approaches. There are many ways one could design a hybrid system, some of them are highlighted below :

- implement content-based and collaborative filtering separately and combine the predictions.
- solve the key limitation of collaborative filtering, the cold-start problem, by designing a collaborative system keeping a track of the item content.
- solve the key limitation of a content-based filtering, the scalability issue, by reducing the dimension of the user data matrix.

- unify the characteristics of both content-based and collaborative filtering through rule-based, probabilistic-based, and knowledge- based techniques/methods.

Table 1.3: Comparison between Recommendation Systems:

Recommender system methods	Content- based filtering	Collaborative Filtering	Hybrid Approach
Number of users	Based on single user	Users based on similar interest	Combination of Content-based & Collaborative filtering
Advantages	User independence, transparency	Improve recommendation performance	Overcome cold start, sparsity problem
Disadvantages	Limited content analysis, New user	Data sparsity, scalability, synonymity	Highly complex, expensive implemetation

1.3 Literature Review

The advent of the "silicon age" has led to the infiltration of technology in every aspect of human life. From using alarm clocks on iPhones to searching through millions and billions of websites through search engines on a daily basis has given rise to an enormous amount of data (big-data). While this enormous quantity of data has supported groundbreaking inventions in the field of AI and machine learning, it also posed a significant challenge. The challenge of finding the most relevant information given a context and criteria in a short period of time. This is where recommendation systems come in, to view and search relevant information with minimal effort.

Technically, recommender systems can be categorized into the following: Content-based Filtering, Collaborative Filtering & Hybrid Approach. Since this work focuses on collaborative filtering based recommender algorithms, the next section outlines a brief literature review of work that has already

been done in this field.

One of the first works on recommendation systems was created in Palo Alto, US at the Xerox Research center in 1992. Back then, there was no efficient process to search through a long list of emails, users had to manually search through those lists to find the emails/content that they required. To address this problem, the researchers at Xerox created Tapestry.[6] The idea was that simply applying a filter would return too many results, so instead returning only the results that were replied to or interacted with at least three or more times works significantly better. This was a manual recommender system. Research went on to create an automated approach to recommending content. In 1994, Paul Resnik and co. in the computer science department at the University of Minnesota developed "GroupLens" [17], GroupLens was developed based on the idea that people's interest would remain unchanged through time, the type of content they liked in the past would mean they are likely to be interested to similar type of content in the future. Researchers behind GroupLens used "NetNews" board to get people to rank news articles on a scale and used those ratings to create groups of people whose ratings are similar to each other. And those groups and ratings to predict how users rate content they haven't seen before.

The utilization of collaborative filtering in the creation of recommender systems meant that users' personal preferences were abstracted away from everyone, leading to a secure way of suggesting content. The commercial deployment of recommender systems soon followed as Amazon.com rolled out its software to its online store in the late 1990s. Customers were suggested new products based on their past interactions, purchases and views. Soon after, multiple companies started integrating recommender systems in their businesses in an effort to increase their sales.

Over the years, these algorithms have been improvised by leaps and bounds owing to the increasing investment from different industries. From the user collaborative filtering method created by GroupLens to item based systems to recently released autoencoder systems. While user based systems generate recommendation for user u for an item i that he/she has not ranked before, based on the assumption that if the users who has similar ranks (neighbors) for other items as u, chances are user u will like it as well. This algorithm faces challenges as the user base grows.

To overcome this, the item based Collaborative filtering algorithm was introduced by Sarwar et. al. [18]. In this paper the authors addressed the problem faced by conventional algorithms of data sparsity (where not enough users have ranked an item for it to be recommended to new users) and scalability as the user base grows rapidly. Their work was focused on analyzing an user-item matrix to produce similarity between items and return recommendations based on those findings. This work was based on the assumption that features between items were relatively static compared to similarity between users - hence they calculated the item-item similarity instead of the users thereby reducing the computing time for millions of users which was a major bottleneck in user based algorithms. The item based algorithm results outperformed the best available user based algorithms of the day.

These models are not very efficient in modeling the changing preferences of users and features overtime. There is a need to address the relative change in the data sets over time. This challenge was addressed by Zuoquan et. al. [14] by creating a probabilistic Hidden Markov Model (HMM).[16] The authors proposed that increasing the past data with subsets of recent data that has been cut off at a point by a time cost parameter from the latest stream of data has the ability to constantly update the dataset with newer item features and user preferences. This model was tested on the MovieLens 100k and Epinions dataset and results were compared to Matrix Factorisation (MF), Time Weight collaborative Filtering (TWCF) and Collaborative Kalman Filter (CKF). The HMM model outperformed MF in the Epinions data, while MF performed marginally better on the MovieLens data. HMM out performed the other two models. Brendan et. al. [10] developed a "Cross-sell" algorithm based on conditionally independent probabilities to provide recommendations in a retail setting. This work addresses the problem of scalability of existing algorithms by pre-computing probabilities of recommendations and using a hash-table look-up process. The novelty of this algorithm lies in the fact that the authors calculated the recommendation probabilities independent of a customer's past purchase history. This comes at a cost of lower accuracy, which is offset by significant reduction of computing power necessary to provide outputs. They reasoned that if a customer has bought a hammer, there is a very high chance that he will buy a nail next. Since this fact is already known, instead of recommending the customer a nail, it's better to recommend him a screwdriver, which will

lead to incremental gains in profit. The authors demonstrated an increase of "profitability" and "re-visit propensity" by 38% and 40% respectively.

A more recent breakthrough in the field of information retrieval and recommender systems is the use of autoencoders. Autoencoders (AE) are a form of unsupervised learning algorithm consisting of an encoder and a decoder. The encoder takes an input and maps the underlying structure in a latent space (the hidden layer) and the decoder takes the data points from the latent space to reconstruct the input with minimal loss of information. [13] These techniques are mostly used for dimensionality reduction tasks in large multi-dimensional datasets. There are several variants of autoencoders that can be found in the literature. **Denoising autoencoder**, this type of AE, the inputs are mixed with noise before passing it to the hidden layer. The output layer (or decoder) attempts to recreate the input from the corrupted input. This forces the network to learn more robust features that generalize well and not overfit [23]. **Stacked denoising autoencoder** (SDAE) uses several denoising AEs to learn a higher level of representations for a more robust training procedure. Major disadvantages include heavy computational costs and lack of scalability [7]. The idea of **Variational autoencoders** (VAE) is to encode the input features as a probability distribution as opposed to random encoding in normal autoencoders. Then the decoder reproduces the input from the latent space using sampling of the encoded data points [9]. A standard autoencoder based recommendation algorithm converts the RS into a binary classification problem. Instead of taking the user ratings directly, the model takes a value of 0 and 1. In practical applications, the user feedback under a predefined threshold is reduced to 0 and 1 otherwise. These models have three layers (input, hidden, output) and use Restricted Boltzmann Machines to train the weights.[24]

Chapter 2

Collaborative Filtering

Algorithms for collaborative filtering can be broadly divided into two main classes: Memory-based and Model-based [1]. In order to generate predictions, Memory-based collaborative filtering utilizes the entire user-item data and uses statistical methods to search for users who have similar pattern/history to the user u. This method is also called nearest-neighbor or user-based collaborative filtering. In contrast, Model-based collaborative filtering uses both explicit and implicit information to develop a model by machine learning techniques such as classification, clustering, and rule-based approach to provide recommendations. Su and Khoshgoftaar [21] stated that model- based approach has better predictions than memory-based since it can handle the challenges of sparsity and scalability better. However, model-based approach requires more resources, such as time and memory in order to develop a model and has the potential to lose information during dimensionality reduction.

2.1 Item-based CF Model

Item-based approach focuses on the set of items the target user u has rated or searched and computes how similar they are to the target item i and then selects n most similar items i_1,i_2,...,i_n and then at the same time their corresponding similarities s_{i1},s_{i2},...,s_{in} are also calculates. Once the most similar items are generated, the prediction is then computed by taking a weighted average of the target user u's ratings on these similar items.

2.2 User-based CF Model

In the user-based approach, recommendations employs statistical techniques to find a set of 'neighbors'. It is used to predict the items, the target user u, might like on the basis of ratings given to that item by other users ('neighbors') who have a history of agreeing with the user u (i.e. either they rate different items similarly or buy similar set of items). Once the neighbourhood set of users is created, the system use different algorithms to combine the preferences of neighbors to generate predictions for the target user u.

2.3 Similarity Measures

The basic idea behind measuring the similarity between two items is to isolate the users who have rated them and to apply a similarity computation technique. Let us consider two items i and j and define there similarity as $s_{i,j}$. A few ways to compute similarity are as follows [18] :

Cosine-based Similarity
Two items are considered to be vectors in m dimensional user-space. There similarity is measured by computing the cosine angle between the two vectors. Cosine-based similarity has been used in this work. Similarity between the items i and j are denoted as :

$$s_{i,j} = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{||\vec{i}||_2 \times ||\vec{j}||_2} \tag{2.1}$$

where "$\cdot$" denotes the dot-product of the two vectors.

Correlation-based Similarity
In this case the similarity is measured by the correlation $corr_{i,j}$ between two items i and j. Let us denote a set of users who have both rated items i and j as U, the correlation similarity is denoted as :

$$s_{i,j} = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_j)^2}} \tag{2.2}$$

Here, $R_{u,i}$ denotes the ratings of the user u and item i, and $\bar{R}_i$ is the average rating of the i-th item.

Adjusted Cosine Similarity
In case of user-based CF the similarity is computed along the rows of the matrix, whereas, in case of item-based CF the similarity is computed along the columns i.e. each pair corresponds to a different user. Computing similarity using basic-cosine measure in item-based approach faces a drawback - the differences in rating scale between different users are not taken into account. The adjusted cosine similarity counters this drawback by subtracting the corresponding user average from each co-rated pair. Similarity between items i and j is denoted as :

$$s_{i,j} = \frac{\sum_{u \in U}(R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U}(R_{u,i} - \bar{R}_u)^2}\sqrt{\sum_{u \in U}(R_{u,j} - \bar{R}_u)^2}} \tag{2.3}$$

Here $\bar{R}_u$ is the average of the u-th user's ratings.

The similarity computations of user-based collaborative filtering is quite like item-based CF, instead of computing the similarity between the items, the focus is on computing the similarity between the users.

2.3.1 Prediction Computation

Item-based CF : The prediction of an item i for a user u is computed by the sum of the ratings given by the user u on items similar to the items i. Each rating is weighted by the corresponding similarity $s_{i,j}$ and the final weighted sum s divided by the sum of similarity to get the normalized prediction value. Let us denote the prediction as $P_{u,i}$:

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,u}|)} \tag{2.4}$$

User-based CF : Let's denote the similarity between two users u and v as $s_{u,v}$ and the prediction of an item i for user u is calculated by computing weighted sum of different users ratings on item i. Let's denote the prediction as $P_{u,i}$:

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v (s_{u,v})} \tag{2.5}$$

where $r_{v,i}$ is the rating of user v on the item i.

Chapter 3

Experiments

3.1 Datasets

In the experiments conducted for this thesis, we will be using 2 datasets :

- **MovieLens 20M Dataset** (Link) - This dataset contains 20000263 ratings and 465564 tag applications across 27278 movies. These data were created by 138493 users between January 09, 1995 and March 31, 2015.
  - It contains six files: movies, ratings, tag, link, genome scores, genome tags :
    - **Movies** : There are three columns in this file - movieId, title and genres. MovieIds ranges from 1 and 131262. There are 27278 unique movie titles and there the 20 unique genres namely 'Romance', 'Musical', 'Fantasy', 'Thriller', 'Children', 'Action', 'Documentary', 'Crime', 'Mystery', 'IMAX', 'Film-Noir', 'Western', 'Sci-Fi', 'Drama', 'Adventure', 'War', 'Animation', 'Comedy', 'Horror' and '(no genres listed)'.
    - **Ratings** : There are four columns in this file - userId, movieId, rating and timestamp. There are 138493 unique userIds. The total number of movieIds are 20000263 with 131262 unique Ids(similar to the Movies file). The ratings range from 0.5 - 5. The average rating is 3.53 and the frequency of the rating 4.0 is the highest followed by 3.0 and 5.0. The timestamp column has 20000263 unix timestamp values.

  - **Tag** : There are four columns in this file - userId, movieId, tag and timestamp. The total number of userIds, movieIds and timestamps are 465564. Unique number of userIds and movieIds are 138472 and 131258 respectively with 38644 unique tags.
  - **Genome Scores** : Genome Scores file contains movie-tag relevance data in three columns - movieId, tagId and relevance.
  - **Genome Tags** : Genome Tags file contains tag description and two columns - tagId and tag.
  - **Links** : The link file contains identifiers that can be used to link to other sources in three columns - movieId, imdbId and tmdbId.

- **Book Recommendation Dataset** (Link)- This dataset contains 242135 unique Book Titles, 102024 unique Authors and 278854 users. These data are obtained from Amazon Web Services.

  - It contains three files: Books, Ratings and Users :

  - **Books** : There are nine columns in this file - ISBN (Books are identified by their respective ISBN), Book-Title (242135 unique values), Book-Author (102024 unique values), Year-Of-Publication, Publisher. URLs linking to cover images are given, appearing in three different sizes (Image-URL-S (271044 unique values), Image-URL-M (271044 unique values), Image-URL-L(271042 unique values)), i.e., small, medium, and large. These URLs point to the Amazon web site.
  - **Ratings** : The ratings file contains three columns - User-ID, ISBN, Book-Rating. The Book-Ratings are either explicit, expressed on a scale from 1-10 (higher values denoting higher appreciation), or implicit, expressed by 0. Books are identified by their respective ISBN with 340556 unique values.
  - **Users** : The Users file contains three columns - User-ID, Location and Age. User-ID have been anonymized and mapped to integers with 278858 unique user ids. Demographic data (Location) and Age of the users are provided (if available) with 57339 unique values. Otherwise, these fields contain NULL-values.

3.1.1 Exploratory Data Analysis

- **MOVIE LENS 20M**

  Figure 3.1 shows a word-cloud visualization of the movie titles. We can recognize that some the most commonly occurring words in the movie titles are Man, Love, Girl, Day, La, Le, de, Night and One.

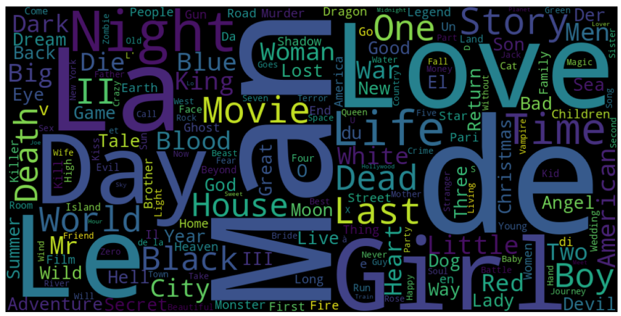


Figure 3.1: Movie Title WordCloud

Figure 3.2 shows a histogram, that show the ratings distribution, we can see that the users are generous and most the ratings are between 3 and 5, with 4 being rated the most. The average rating is 3.53.

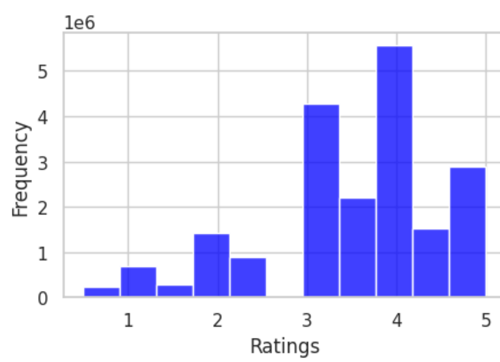


Figure 3.2: MovieLens 20M ratings frequency

Figure 3.3 shows the list of top 10 most rated movies with the count on the left. Figure 3.4 (a) shows the list of the genres with it's counts on the right, we can see Drama, Comedy, Thriller, Romance and Action

```
Count: 67310, Pulp Fiction (1994)
Count: 66172, Forrest Gump (1994)
Count: 63366, Shawshank Redemption, The (1994)
Count: 63299, Silence of the Lambs, The (1991)
Count: 59715, Jurassic Park (1993)
Count: 54502, Star Wars: Episode IV - A New Hope (1977)
Count: 53769, Braveheart (1995)
Count: 52244, Terminator 2: Judgment Day (1991)
Count: 51334, Matrix, The (1999)
Count: 50054, Schindler's List (1993)
```

Figure 3.3: Top 10 most rated movies

```
Genres and their counts:
Drama                 13344
Comedy                 8374
Thriller               4178
Romance                4127
Action                 3520
Crime                  2939
Horror                 2611
Documentary            2471
Adventure              2329
Sci-Fi                 1743
Mystery                1514
Fantasy                1412
War                    1194
Children               1139
Musical                1036
Animation              1027
Western                 676
Film-Noir               330
(no genres listed)      246
IMAX                    196
dtype: int64
```

(a) Genre count

(b) Genre WordCloud

Figure 3.4: Movie Genres

are the top genres of the movies in the MovieLens 20M Dataset. Similarly in 3.4 (b) we can see a genre wordcloud based on it's frequency.

In Figure 3.5, the bar plot illustrates the distribution of released movies across different years. Notably, there was a significant surge in movie releases during the years 2000 and 2005. However, subsequent years have witnessed a gradual decline in the number of movies being released. According to a news article by CNBC [4] it might be because since 2000 to 2005 DVD sales reached $16.3 billion. Leading to an economic downturn there was a 3% drop in 2006 and since 2007, DVD sales actually rose about half a percent. A combination of the Great Recession, a rise in customers buying on-demand and digital copies of films and the launch of streaming services is what led to gradual decline

in the number of movies being released.

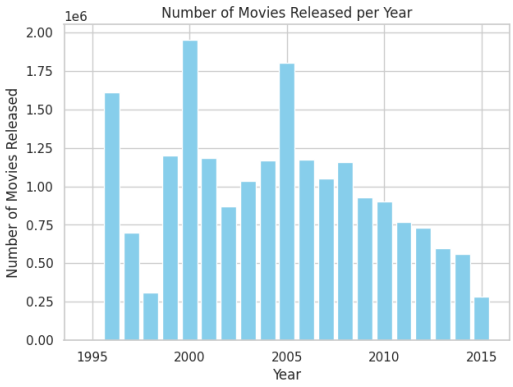


Figure 3.5: Movies released based on Years

- **BOOK RECOMMENDATION**

Figure 3.6 shows a word-cloud visualization of the book titles. We can see that some of the most commonly occuring words are Book, Novel, Stories, Love, Guide, Series, Time, Women, Heart, Story.

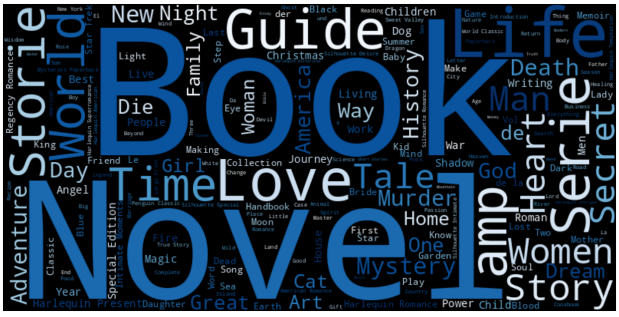
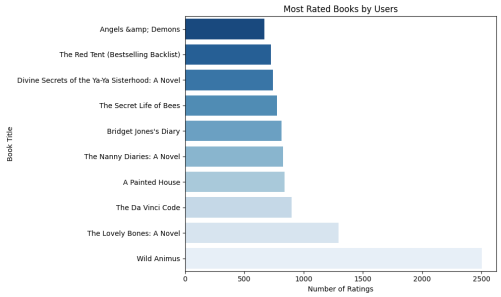


Figure 3.6: Book Titles WordCloud

Figure 3.7 (a) and (b) suggests that the top-rated books often belong to the fiction genre and have engaging stories with well-developed characters. Many of these books are written by famous authors and have been marketed effectively. This shows that a captivating plot, relatable characters, author reputation, and good marketing are important factors in making a

	Book-Title	Book-Rating
8	The Secret Life of Bees	8.477833
4	The Da Vinci Code	8.439271
5	The Lovely Bones: A Novel	8.185290
7	The Red Tent (Bestselling Backlist)	8.182768
1	Angels & Demons	8.016129
3	Divine Secrets of the Ya-Ya Sisterhood: A Novel	7.876161
2	Bridget Jones's Diary	7.625995
6	The Nanny Diaries: A Novel	7.437659
0	A Painted House	7.398907
9	Wild Animus	4.390706

(a) Top 10 Book's average rating

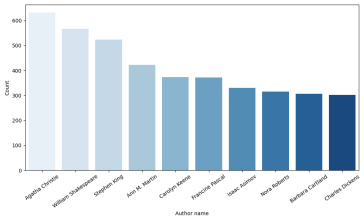


(b) Most Rated Books

Figure 3.7: Books

book highly rated.

Figure 3.8 (a) and (b) shows the lists of top-rated authors based on their average ratings reveals several noteworthy observations of the dataset. Renowned playwright William Shakespeare stands out as the highest-rated author, likely due to his enduring literary contributions. Prominent figures in the world of fiction, such as Isaac Asimov, Agatha Christie, and Stephen King, also make the list, underlining the enduring popularity of their imaginative storytelling and mastery of their respective genres. Additionally, writers such as Carolyn Keene and Nora Roberts are known for their prolific output.Collectively, these authors' enduring impact, genre expertise, and prolific writing contribute to their status as the top-rated authors in this dataset.



(a) Top 10 rated authors

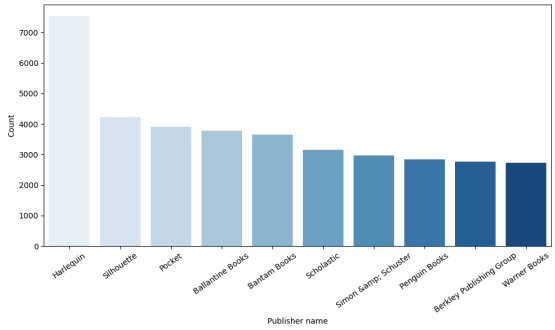
	Book-Author	Book-Rating
9	William Shakespeare	8.303357
6	Isaac Asimov	7.915167
3	Carolyn Keene	7.823529
8	Stephen King	7.815046
0	Agatha Christie	7.792102
4	Charles Dickens	7.762836
7	Nora Roberts	7.629680
1	Ann M. Martin	6.708333
5	Francine Pascal	6.659091
2	Barbara Cartland	5.041812

(b) Top 10 authors - Average rating

Figure 3.8: Book Authors

	Publisher	Book-Rating
6	Scholastic	7.818161
4	Penguin Books	7.773789
0	Ballantine Books	7.610437
5	Pocket	7.564115
9	Warner Books	7.543623
8	Simon & Schuster	7.522894
1	Bantam Books	7.520811
2	Berkley Publishing Group	7.443836
7	Silhouette	7.239616
3	Harlequin	7.170115

(a) Top 10 publishers



(b) Top 10 publishers-Average rating

Figure 3.9: Book Publishers

Figures 3.9 (a) and (b) shows the top 10 publishers in the dataset and their average rating by the users. Scholastic and Penguin Books lead with the highest average ratings of 7.82 and 7.77, respectively, likely due to their strong reputations and diverse book offerings. Ballantine Books and Pocket follow closely with average ratings of 7.61 and 7.56, Warner Books, Simon & Schuster, and Bantam Books maintain competitive ratings around 7.52. Berkley Publishing Group, Silhouette, and Harlequin round out the top 10 with slightly lower average ratings, but they still demonstrate a commitment to providing quality literature.

3.2 Collaborative Filtering Experiments

This work trained and tested two recommendation system algorithms : user-based and item-based CF models on two different datasets to explore the performance in recommending users relevant content. First, the Movie Lens-20 million dataset was used. To reduce compute overhead due to a lack of GPU-based cloud computing resources, the first 100,00,00 rows were used. The entire dataset was first downloaded from Kaggle and loaded into a pandas dataframe, where only the subset of a dataset was selected for training and testing of the models. The code snippet (Figure A.1: Code Snippet - 1) shows the data_loader function that reads the CSV file into a dataframe and returns it.

The function is nested inside the utils folder, and is called from "user_based_CF.py" and "item_based_CF.py"

The snippets contain two more utility functions: one to return the dataframe containing the movie.csv file and the other (movie_recommendation) that takes a dictionary (Figure A.2: Code Snippet - 2) and returns the movie names based on those IDs from the movie.csv file. (Figure A.3: Code Snippet - 3)

The movie_recommender function (Figure A.4: Code Snippet - 4) can be found in the recommend.py script nested inside the same utils folder.

Now, on to the CF models. The models are written using the LibRecommendation library. This is an open source recommender system library that provide TensorFlow, PyTorch, Cython implementations of major recommender algorithms in the industry Link.

ITEM BASED MODEL - MOVIE LENS 20M

In this section, the item based CF model implementation is explained. The code snippet (Figure A.5: Code Snippet - 5) illustrates the necessary library imports that are used throughout the script.

The random_split functions splits the dataset into train and test sets randomly based on provided ratios and the DatasetPure class creates an object that is used to feed the model with the relevant data during the training and testing processes. The userCF class compiles the model object.

kaggle_data_loader and movie_recommender are written by the author of this work to help load the data and map movie IDs returned by the model to its corresponding movie titles.

The code snippet (Figure A.6: Code Snippet - 6) illustrates the data loading and processing stage followed by the creation of the item based model object.

The data is first loaded in and the first 1 million rows are selected. The column names are renamed to comply with the naming requirements of the library, and split into a ratio of 80% train set and 20% test set. The Dataset-

Pure class creates a train_data object to feed into the model, if any other features were used (age, height, gender etc) to create more complex models, DatasetFeat had to be used. Compute constraints influenced the decision to build a simpler version of the model with only user, item and label columns.

The ItemCF class compiles the model and expects the following values as constructors:

"Task": Two options are provided by the library, rating and ranking. In this work rating is used, where the model predicts the possible rating an user might provide, making it a regression problem. This influenced the choice of metrics to evaluate the trained model, mean squared error (MSE), root mean squared error (RMSE) and mean absolute error (MAE). Explanations for these are provided in the **Experimental Results** section. The evaluate function is fed with the test set on which the model predicts ratings of users from a unseen set of data and is compared with the ground truth values.

"data_info": The data_info is returned by the build_trainset method of the DatasetPure class of the library. This object returns a dictionary which contains information about the unique number of users and movies, and the sparsity of the data expressed in percentage. For the data that is used in this work, it looks like this on the terminal (Figure A.7: Code Snippet - 7)

"sim_type": The cosine similarity is used and have been observed to perform best for this problem. Cosine similarity is described in section **cosine similarity**

"k_sim": The number of similar items to consider.

"store_top_k": Takes a boolean value to decide whether to store the top K similar items in memory.

The snippet (Figure A.8: Code Snippet - 8) shows the model.fit method where the model is trained on the data, and tested on the set of unseen data. Loss, MSE, and MAE metrics are stored and returned after the testing, RMSE is calculated afterwards. The results of this code are discussed in more detail in the **Experimental Results** section.

The recommend_user method of the model class returns a dictionary of recommendations which is then mapped to the corresponding movie titles.

After the training was completed, the model outputted the following recommendations shown in Figure A.9: Code Snippet - 9 .

As shown in the snippet, the user has given a rating of 5 to Jumanji which falls under the Adventure, Children and Fantasy genre and hence the model recommended 10 movies which falls under one or more of those categories or is closely related in genres.

USER BASED MODEL - MOVIE LENS 20M

This section illustrates the implementation of the user based model. Since the same library and code structure and utility functions are used, only the minor differences in the code will be highlighted in this section (Figure A.10: Code Snippet - 10).

The UserCF algorithm is imported from libreco.algorithms module and the user_based_model object is created. The rest of the script is same as before. The model is tested on the same user to check if the suggestions are varying than the item based model. These are shown in Figure A.11: Code Snippet - 11 .

The movie recommendations are quite different than the ones suggested by the item based model with the recommendations inclining more towards the "crime" genre, which may or may not be very appropriate for a user depending on his/her age. The evaluation metrics will be discussed in a later chapter which will quantify the performance of both of these models on the dataset.

ITEM BASED MODEL - BOOK RECOMMENDATION DATASET

The book recommendation dataset was collected from Kaggle which contains 1 million rows of data containing book IDs (ISBNs), User IDs, Book title, user rating etc. The layout of the script is exactly the same and is written with the LibRecommender library. The code snippet (Figure A.12: Code Snippet - 12) outlines the data loading part of the script.

The relevant columns names are changed to conform with the model library requirements. ISBN is an unique identifier for every book which in this context works as the item, the book-rating column contains predictions in the range 0 - 10, classified as label. User-ID column identifies users and is changed to user. Users with rating of 0 are dropped as they are implicit feedback, (indicating clicks, interaction with the book but no direct rating provided). These do not provide any benefit to the model, and the model performed much better without them as found during experiments with both the implicit ratings in and out (3.04 MAE when included and 1.35 MAE when excluded). .

The dataset is randomly split into train and test set in the ratio of 80% to 20%. The model object is created to perform a rating task and consider 10 similar items. Similarity is calculated using cosine similarity (Figure A.13: Code Snippet - 13).

Following the model training and evaluation, the recommend_user method was used to ask for 10 book recommendations for user 276786. The code snippet (Figure A. 14: Code Snippet - 14) illustrates the output alongside the user's previous ratings.

As outputted by the model, the user previously rated a Vieja Nueva York a 0 / 10 indicating that he/she probably did not enjoy the book at all. The model suggested him 10 books of different genres (the book that the user rated was Fiction) for example "The other daughter" falls under Mystery, Thriller and Fiction, while "Waiting for Nick" falls under romance and fiction (source: Google) and so on. The metrics on the test set that was collected from the evaluate method, will be discussed in the **Experimental results** section.

USER BASED MODEL - BOOK RECOMMENDATION DATASET

The user based CF model follows the same code with minor differences, for example: userCF class is used instead of itemCF inside the libreco.algorithms module. This section highlights the output and recommendations of the model to see if there are any notable differences with the item based CF model (Figure A.15: Code Snippet - 15).

The recommendations for the userbased CF model as seen are a bit different for the same user. Although the books all incline towards thriller mystery and guide/lifestyle. The following section goes in depth into quantifying the performance of the two models over the two datasets.

3.3 Experimental Results

Assessing model performance is paramount to guarantee precise and valuable suggestions for users to provide insights into the accuracy and quality of its predictions. In this context, we explore three key evaluation metrics: Mean Squared Error, Root Mean Squared Error, and Mean Absolute Error.

Mean Squared Error (MSE) :

MSE measures the average squared difference between actual ratings and predicted ratings to penalizes larger errors more heavily, giving a higher weight to outliers.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\text{actual\_rating}_i - \text{predicted\_rating}_i)^2$$

Root Mean Squared Error (RMSE) :

RMSE is the square root of MSE and it provides the average magnitude of the errors in the predicted ratings emphasizing the overall prediction quality providing a meaningful value in the same scale as the original ratings.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

Mean Absolute Error (MAE) :

MAE calculates the average absolute difference between actual and predicted ratings. Unlike MSE and RMSE, MAE doesn't square the errors, therefore, overestimates and underestimates equally. We use MAE to focus on the accuracy of individual predictions without being disproportionately influenced by outliers to achieve more robust indication of the model's overall performance by considering the average magnitude of errors.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |\text{actual\_rating}_i - \text{predicted\_rating}_i|$$

In this section, the results of the recommendation systems based on item-based and user-based approaches are presented which are conducted on two distinct datasets - Move Lens - 20M and Book recommendation.

Table 3.1: Evaluation Metrics: MovieLens-20M

Metric	User-based	Item-based
MSE	8.797	0.758
RMSE	2.966	0.871
MAE	2.859	0.657

Table 3.2: Evaluation Metrics: Book Recommendation

(a) With implicit rating (rating 0 included)

Metric	User-based	Item-based
MSE	18.922	14.622
RMSE	4.350	3.824
MAE	3.586	3.047

(b) Without implicit rating (0 rating excluded)

Metric	User-based	Item-based
MSE	4.008	3.261
RMSE	2.002	1.806
MAE	1.562	1.359

In the Movie Lens dataset, the user-based recommendation system performs reasonably well, with an RMSE of 2.967 and MAE of 2.859. However, the item-based approach significantly outperforms it, achieving a much lower RMSE (0.872) and MAE (0.657). In the Books Recommendation System, the item-based recommendation system outperforms the user-based approach in terms of prediction accuracy. The item-based model achieves a lower RMSE (3.824) and MAE (3.047) compared to the user-based model with an RMSE of 4.351 and MAE of 3.586.

Through the comparative analysis we can see that, in both datasets, the item-based recommendation system outperformed the user-based system. There can be several reasons why the item-based approach worked better than the user-based approach. For both books and movies, items have inherent characteristics that can be used for similarity calculations. For books, attributes like genre, author, and theme are readily available and can be used to identify similar items. Similarly, movies can be categorized based on genres, directors, and actors, providing clear item-item relationships hence items have a consistent and comprehensive data, making it easier to establish meaningful similarities among them, whereas due to user data sparsity, i.e. the users have rated a fraction of the movies(or books) in the dataset, for example, a user might have rated a few movies but left most of their watched movies unrated, which makes finding similar users with adequate overlap in preferences challenging, leading to less reliable suggestions. Another significant reason can be that the users' tastes and preferences vary widely, making it challenging to find similar users with matching preferences, while, items typically have more consistent characteristics, making the results more reliable. Item-based approach excels at recommending niche or "long-tail" items that have fewer user interactions but might be highly relevant to specific users which is not the case for the user-based approach. Lastly, unlike user-based systems, item-based systems scaled well and handled the 'cold start' problem more effectively.

Chapter 4

Conclusion

In this work, collaborative filtering techniques for recommendation systems were examined. Two models, item-based and user-based systems were explained and implemented in python, and tested against two publicly available data - movie lens and books recommendations. Across the datasets, the item-based model outperformed the user based model, supporting the reports from the existing literature. It was also seen that excluding implicit feedback data points improved performance in both the models in the book recommendations dataset. Although the nature of the recommendations for a randomly selected user was found to be similar, the quantitative metrics indicated the superiority of the item-based models in both the datasets. Furthermore, it was also noticed that using a subset of data (1M in movie lens 20M) to train the model significantly improved training time, with negligible loss in performance, proving that CF models don't always require endless rows of user/item data to perform well. A deep dive analysis was also provided in order to understand the underlying nuances of these datasets, and process them for training a ML model. There still remain a few pathways that can be taken to explore further on the basis of this work. First would be to include user meta-data in the train set of the model, to see if more information in the data leads to better inference. Several advanced algorithms are in use in the industry today (some of them briefly touched upon in the literature review section) and a deeper exhaustive study of these models can be done to get a comprehensive view of the state-of-the-art in the field of recommendation systems.

Bibliography

[1] PH Aditya, Indra Budi, and Qorib Munajat. A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for e-commerce in indonesia: A case study pt x. In *2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pages 303–308. IEEE, 2016.

[2] Nicholas J Belkin and W Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[3] Kurt D Bollacker, Steve Lawrence, and C Lee Giles. Citeseer: An autonomous web agent for automatic retrieval and identification of interesting publications. In *Proceedings of the second international conference on Autonomous agents*, pages 116–123, 1998.

[4] CNBC. The death of the dvd: Why sales dropped more than 86 `https://www.cnbc.com/2019/11/08/the-death-of-the-dvd-why-sales-dropped-more-than-86percent-in-13-years.html`, 11 2019.

[5] Zeshan Fayyaz, Mahsa Ebrahimian, Dina Nawara, Ahmed Ibrahim, and Rasha Kashef. Recommendation systems: Algorithms, challenges, metrics, and business opportunities. *Applied Sciences*, 10(21), 2020.

[6] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[7] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[8] Folasade Olubusola Isinkaye, Yetunde O Folajimi, and Bolande Adefowoke Ojokoh. Recommendation systems: Principles, methods and evaluation. *Egyptian informatics journal*, 16(3):261–273, 2015.

[9] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[10] Brendan Kitts, David Freed, and Martin Vrieze. Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 437–446, 2000.

[11] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: Recommendation models, techniques, and application fields. *Electronics*, 11(1), 2022.

[12] Madhusree Kuanr and Puspanjali Mohapatra. *Recent Challenges in Recommender Systems: A Survey*, pages 353–365. 01 2021.

[13] Pengzhi Li, Yan Pei, and Jianqiang Li. A comprehensive survey on design and application of autoencoder in deep learning. *Applied Soft Computing*, page 110176, 2023.

[14] Zuoquan Lin and Hanxuan Chen. A probabilistic model for collaborative filtering. In *Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics*, WIMS2019, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Miquel Montaner, Beatriz López, and Josep Lluís De La Rosa. A taxonomy of recommender agents on the internet. *Artificial intelligence review*, 19:285–330, 2003.

[16] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[19] Markus Schedl, Hamed Zamani, Ching-Wei Chen, Yashar Deldjoo, and Mehdi Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7:95–116, 2018.

[20] Fabio Soldo, Anh Le, and Athina Markopoulou. Predictive blacklisting as an implicit recommendation system. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[21] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

[22] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research*, 10:623–656, 2009.

[23] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[24] Nan Zhang, Shifei Ding, Jian Zhang, and Yu Xue. An overview on restricted boltzmann machines. *Neurocomputing*, 275:1186–1199, 2018.

[25] Tao Zhou, Zoltán Kuscsik, Jian-Guo Liu, Matúš Medo, Joseph Rushton Wakeling, and Yi-Cheng Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515, 2010.

Appendix A

Appendix

A.1 System

Programming Language: Python == 3.9.13

Library dependencies:

LibRecommender == 1.3.0
Pandas == 1.5.3

System: Processor:

10th generation Intel core i7
RAM: 40 GB
Model trained on CPU.

A.2 Running Instructions:

Open the code folder, launch in VS Code, and open a terminal from inside the code folder. In the terminal type the following to run the script. This code assumes the user is inside the code folder but outside the individual dataset folder. Dependencies need to be installed once per virtual environment. Recommended running on python 3.9.13 on which the code was written, library versions are provided in the appendix section A.1.1. Training and prediction time will vary depending on system specifications. Data analysis notebooks are provided in the code folder, and can be run separately.

For movie lens:

- cd movieLensCFModels
- pip install -r requirements.txt
- python -m user_based_CF
- python -m item_based_CF

For book recommendations:

- cd bookRecommendationsCFModels
- python -m user_based_CF
- python -m item_based_CF

A.3 Code Snippets :

```
import pandas as pd

def kaggle_data_loader():
      rating_df = pd.read_csv('data_kaggle/rating.csv')
      return rating_df
```

Figure A.1: Code Snippet - 1

```
{ userID : np.array(movieIDs) }
```

Figure A.2: Code Snippet - 2

```
def kaggle_data_loader_recommend():
    movie_df = pd.read_csv('data_kaggle/movie.csv')
    return movie_df
```

Figure A.3: Code Snippet - 3

```
import pandas as pd
from data_loader import kaggle_data_loader,
kaggle_data_loader_recommend

def movie_recommender(recommendation_dict):
    rating_df = kaggle_data_loader()
    movie_df = kaggle_data_loader_recommend()
    merged_df = pd.merge(rating_df, movie_df, on='movieId')

    for user in recommendation_dict:
        print(f'Movie recommendations for user: {user}')
        for movie_id in recommendation_dict[user]:
            print(merged_df.loc[merged_df['movieId'] ==
movie_id, 'title'].iloc[0])
```

Figure A.4: Code Snippet - 4

```
# imports
import numpy as np
import pandas as pd
from libreco.data import random_split, DatasetPure
from libreco.algorithms import UserCF
from libreco.evaluation import evaluate
from utils.data_loader import kaggle_data_loader
from utils.recommend import movie_recommender
```

Figure A.5: Code Snippet - 5

```
# load data
    movie_lens_data = kaggle_data_loader()
    movie_lens_data = movie_lens_data.head(1000000)
    movie_lens_data = movie_lens_data.rename(columns={
        "userId": "user",
        "movieId": "item",
        "rating": "label"
    })

    # split dataset into training and testing data
    train_data, test_data = random_split(
        movie_lens_data, multi_ratios=[0.8, 0.2]
        )

    # create train data object and extract data info
    train_data, data_info = DatasetPure.build_trainset(
        train_data
    )

    # test data
    test_data = DatasetPure.build_testset(
        test_data
    )

    print(data_info)

    # compile item-based model
    item_based_model = ItemCF(
                            task="rating",
                            data_info=data_info,
                            sim_type="cosine",
                            k_sim=10,
                            store_top_k=True
                            )
```

Figure A.6: Code Snippet - 6

```
n_users: 6743, n_items: 13359, data density: 0.8881 %
```

Figure A.7: Code Snippet - 7

```
   # train model
    item_based_model.fit(
        train_data,
        verbose=2,
        neg_sampling=False,
        metrics=["loss", "rmse", "mae"]

)

    # evaluate model performance on test set
    test_metrics =  evaluate(
        model = item_based_model,
        data = test_data,
        neg_sampling=False,
        metrics=["loss", "rmse", "mae"]
    )

    # print metrics
    print(f"Test metrics: {test_metrics}"
)
```

Figure A.8: Code Snippet - 8

```
user 296 has previously rated: title Jumanji (1995)
rating 5.0
genres    Adventure|Children|Fantasy
Name: 38, dtype: object
Movie recommendations for user: 296
title                Jurassic Park (1993)
genres    Action|Adventure|Sci-Fi|Thriller
Name: 2632192, dtype: object
title             Batman (1989)
genres    Action|Crime|Thriller
Name: 7867739, dtype: object
title          Forrest Gump (1994)
genres    Comedy|Drama|Romance|War
Name: 4789224, dtype: object
title               Speed (1994)
genres    Action|Romance|Thriller
Name: 4886719, dtype: object
title     Clear and Present Danger (1994)
genres        Action|Crime|Drama|Thriller
Name: 7696932, dtype: object
title     Die Hard: With a Vengeance (1995)
genres                Action|Crime|Thriller
Name: 4739451, dtype: object
title                                Lion King, The (1994)
genres    Adventure|Animation|Children|Drama|Musical|IMAX
Name: 5318019, dtype: object
title     Mrs. Doubtfire (1993)
genres             Comedy|Drama
Name: 5382844, dtype: object
title            Stargate (1994)
genres    Action|Adventure|Sci-Fi
Name: 3176052, dtype: object
title     Terminator 2: Judgment Day (1991)
genres                       Action|Sci-Fi
Name: 535540, dtype: object
```

Figure A.9: Code Snippet - 9

```
from libreco.algorithms import UserCF

 # compile user-based model
 user_based_model = UserCF(task="rating",
                           data_info=data_info,
                           sim_type="cosine",
                           k_sim=10,
                           store_top_k=True
                           )
```

Figure A.10: Code Snippet - 10

```
user 296 has previously rated: title  Jumanji (1995)
rating 5.0
genres    Adventure|Children|Fantasy
Name: 38, dtype: object
Movie recommendations for user: 296
title     Mary Shelley's Frankenstein (Frankenstein) (1994)
genres                                  Drama|Horror|Sci-Fi
Name: 13078154, dtype: object
title     Léon: The Professional (a.k.a. The Professiona...
genres                          Action|Crime|Drama|Thriller
Name: 296269, dtype: object
title         Tombstone (1993)
genres    Action|Drama|Western
Name: 6092314, dtype: object
title     Usual Suspects, The (1995)
genres        Crime|Mystery|Thriller
Name: 118992, dtype: object
title     Quiz Show (1994)
genres               Drama
Name: 9861953, dtype: object
title        Babe (1995)
genres    Children|Drama
Name: 9748275, dtype: object
title     Seven (a.k.a. Se7en) (1995)
genres               Mystery|Thriller
Name: 75743, dtype: object
title     Braveheart (1995)
genres     Action|Drama|War
Name: 2559897, dtype: object
title     Shawshank Redemption, The (1994)
genres                          Crime|Drama
Name: 389383, dtype: object
title     Star Trek: Generations (1994)
genres          Adventure|Drama|Sci-Fi
Name: 3207851, dtype: object
```

Figure A.11: Code Snippet - 11

```
# load data
books_data = data_loader_rating()
books_data = books_data.rename(columns={
    "User-ID": "user",
    "ISBN": "item",
    "Book-Rating": "label"
})

books_data = books_data.drop(books_data.query('label == 0').index)
```

Figure A.12: Code Snippet - 12

```
# compile item-based model
item_based_model = ItemCF(
                        task="rating",
                        data_info=data_info,
                        sim_type="cosine",
                        k_sim=10,
                        num_threads=3,
                        store_top_k=True
                        )

# train model
item_based_model.fit(
    train_data,
    verbose=2,
    neg_sampling=False,
    metrics=["loss", "rmse", "mae"]
)
```

Figure A.13: Code Snippet - 13

```
user 276786 has previously rated: Book-Title     Vieja Nueva York
Book-Rating                    0
Name: 1931, dtype: object
Book recommendations for user: 276786
The Hamlyn complete knitting course
KÃ?Â1/4sschen, KÃ?Â1/4sschen.
Das WÃ?Â1/4ten der ganzen Welt.
Home Freezing of Fruits and Vegetables
Considering Kate (The Stanislaskis) (Silhouette Special Edition)
Waiting For Nick (Silhouette Special Edition)
The Other Daughter
The Next Accident
```

Figure A.14: Code Snippet - 14

```
user 276786 has previously rated: Book-Title     Vieja Nueva York
Book-Rating                    0
Name: 1931, dtype: object
Book recommendations for user: 276786
For Love
Wildfire at Midnight
The Shadows of Power
Get in Shape the Lazy Way (Macmillan Lifestyles Guide)
```

Figure A.15: Code Snippet - 15