Final Year Project Report

Evaluating Recommender Systems

Congcong Wang

BSc. (Hons.) in Computer Science

Supervisor: Micheal O'Mahony

UCD School of Computer Science
University College Dublin

October 17, 2018

Project Specification

General Information:

Recommender systems assist consumers in discovering information, products and services that are relevant to their needs. Recommender systems are now widely deployed in many settings and many of us routinely consume recommendations from the likes of Amazon, Netflix, and Last.fm, for example.

In order for recommender systems to be effective, the relevance of recommendations made to end-users is key. However, other performance criteria are also important; for example, the ability of the recommender system to make novel and diverse recommendations, and to provide high coverage from a catalogue and user perspective, to name a few.

The aim of this project is to develop a Web application to facilitate the offline evaluation of recommender system algorithms. The framework should facilitate the evaluation of different algorithm/dataset/evaluation design combinations. The application should also facilitate the comparison of multiple algorithms according to specified evaluation criteria.

Note that a key requirement of this project involves the implementation of a Web application to evaluate algorithm performance. As such this project is suitable for those with a strong interest in recommender systems and Web application development.

Core:

- Background reading.
- Implementation of three recommendation algorithms. Open-source code may be used for this task.
- Design and development of a Web application to evaluate and compare the offline performance of these algorithms. The application should initially support basic evaluation criteria (relevance, novelty, diversity and user coverage).
- The application should be readily configurable to include additional recommendation algorithms and evaluation metrics.

Advanced:

- Compare and contrast algorithm performance using the application (evaluation datasets will be provided).
- Extend the application to include additional evaluation metrics and algorithms.
- Enhancement(s) of the student's choice.

Abstract

With the increasing amount of information that people browse daily, how to quickly obtain Information Items that meet people's needs has become an urgent issue these days. The effort in information retrieval have brought great convenience to people who tend to retrieve information by entering a query or keywords. If some information-intensive websites can proactively suggest products or information items that users may be interested in, it will greatly improve the efficiency and satisfaction of users in obtaining information items. The research in the field of recommender systems precisely originated on this subject. Over the past years, tremendous progress has been made into this area, from non-personalised to personalised and to more recent deep learning recommender systems. Although recommender systems have been widely applied, there are still many issues and challenges in designing high quality recommender systems. To measure the quality of a recommender system, a scientific and rigorous evaluation process is required. This report reviews some existing well-established recommender systems and investigate some existing metrics for evaluating them. Besides, this report gives details of the project's implementation - a web application for the offline evaluation of three major collaborative filtering recommendation algorithms, item-based, user-based, and matrix-factorisation. The application supports a wide range of readily configurable evaluation metrics for users to visualise the performance between different recommendation algorithms. The project aims to provide a comprehensive platform for designers to evaluate recommender systems and guide them to design better recommender systems.

Acknowledgements

This project is aided by my supervisor Micheal O'Mahony. I would like to thank him not only for his guidance and patience, but also for his continuous suggestions and feedbacks to this work.

Table of Contents

Chapter 1: **Introduction**

1.1 Motivation

Nowadays, a recommender system can be found in almost every information-intensive website. For example, a list of likely preferred products are recommended to an customer when browsing the target product in Amazon[1] [21]. Moreover, when watching a video clip in Youtube[2], a recommender system employed in the system [52] suggests some relevant videos to users by learning the users' behaviours that were generated previously. So to speak, recommender systems have deeply changed the way we obtain information. Recommender systems not only make it easier and more convenient for people to receive information, but also provide great potential in the economic growth as described in [1]. As more and more people realise the importance and power of recommender systems, the exploration for designing high quality recommender systems have been remaining an active topic in the community over the past decade. Due to the continuous efforts into the field, thankfully, many recommender systems have been developed and used in a variety of domains [31]. Based on this, a key question arising is how to know the performance of recommender systems so that the most suitable ones can be found to apply in certain contexts or domains.

The answer goes to evaluating recommender systems by conducting rigorous and scientific evaluation experiments [7, 3, 8, 52]. Evaluating recommender systems has become increasingly important with the growing popularity of recommender systems in some applications. It is often the case that an application designer need to choose between a set of candidate recommendations algorithms. This goal can be achieved by comparing the performance of these algorithms in evaluation experiments. Besides, evaluating recommender systems can help researchers to select, tune and design recommender systems as a whole. This is because when designing a recommender system, some key factors influencing the system's quality are often come into notice in the process of evaluation. For example, in [7], Herlocker et al. highlight that, when considering evaluation metrics, evaluators should not only take into account the accuracy metrics, but also some extra quality metrics, or to say beyond accuracy metrics, which attach importance to the fact that users are often not interested in the items that they already know and surely like but sometimes in discovering new items and exploring diverse items.

Motivated by the importance of evaluating recommender systems and the highlight on comprehensive metric considerations into the evaluation experiments, the project aims to explore the most possible scientific method to evaluating recommender systems. The implementation of the project is presented in the form of a web application.

[1]https://www.amazon.com/
[2]https://www.youtube.com/

1.2 Project scope

Concerning the project scope, here are several points to emphasise

- **Comprehensive**. It is important to include fuller evaluation metrics especially beyond accuracy metrics [8, 7] when developing the evaluation platform. Here the word 'comprehensive' mainly stresses that the platform should support a well-rounded selection of evaluation metrics. Also, it stresses that the metrics should be available for the compare and contrast between multiple experiments.
- **Offline evaluating**. There are main two approaches to evaluate recommender systems. Online trials uses A/B testing to evaluate different algorithms by analysing real usage and recommendation feedback [25]. Offline evaluation attempts to test core recommendation algorithm components by using real-world rating datasets [7]. The scope of this project is offline evaluation.
- **Collaborative filtering algorithms**. Although a wide range of recommendation algorithms are well-established and applied, the collaborative filtering (CF) algorithms [3, 2, 7, 4, 5, 11] are considered the most widely-used ones. Although the project concentrates on CF algorithms, the application developed is extensible and can readily support the addition of new RS algorithms.

1.3 Objectives and Implementation

Overall, the project aims to provide a good platform for designers to evaluate recommender systems and guide them to design better recommender systems. Based on the project's specification, there are main five objectives extracted. Blow is the brief description of each projective.

1. **A web application**. A web application is developed to provide GUI interfaces for users to conduct experiments more easily and conveniently. The implementation process goes from prototyping, to the design of pages, to coding, and finally to optimisation. Along the way, the key idea that has been bearing in mind is making the front interfaces as easily interactive as possible.
2. **Implementation of three algorithms**. The three algorithms are all collaborative filtering ones, user-based, item-based and matrix factorisation. The implementation is aided by an open-source library named LibRec[3]. The work that need to do is to extend the library and integrate it with the web environment.
3. **Evaluation visualisation**. This objective can be said the core part of the project. First, three evaluation methodologies [32] should be integrated to the system. They are repeated random sub-sampling, leave-one-out and K-fold cross validation. Next, some basic evaluation metrics (here primarily accuracy metrics) are included. Last, the compare and contrast, namely metrics visualisation between multiple experiments should be finished.
4. **Session mechanism**. The system is developed to support two different modes. The first one is that a user can get access to the application as a guest which means there is no

[3]https://www.librec.net

session between the user and the server. The other mode is session mode, which allows users to log in the system using a unique session code. In this mode, users are in session services provided by the server. This objective is outside of the project specification. In consideration of the fact that evaluating a recommender system is often a time-consuming task, the session mechanism is used to manage experiments and store users' running data including the information of experiments, running times, metric details, etc. Core activities involved in the implementation include database design, management and representation of experiments.

5. **Extension and enhancement**. The objective is considered the advanced improvement for the project. The first extension part falls into the dataset. A file uploading interface is provided for users to apply dataset in a certain format. In addition, the most important extension goes to beyond accuracy metrics. This is what the project core implementation lies on. Beyond accuracy metrics that are extended and enhanced include popularity, diversity, novelty, coverage, etc.

The objectives above are listed quite generally and more details are provided in Chapter 4 and 3.

1.4 Report Structure

- *Chapter 2* gives a background of relevant research to the project. A general introduction to recommender systems is described first, including the power and dimensions of recommender systems. Then a commonly-acknowledged taxonomy of recommendation algorithms is introduced briefly, including content-based, hybrid, etc [28]. The focus of this chapter goes to the review of evaluation metrics for evaluating recommender systems.
- *Chapter 3* gives the specific implementations of three collaborative filtering algorithms, user-based, item-based and matrix-factorisation. Some technical points about them are detailed in this chapter.
- *Chapter 4* first gives the system architecture where the core components of the system are described. Second, the chapter presents the details of design thinking and implementation of the project from two aspects, front-end and back-end respectively. Last, the methodology to test the system's performance is presented.
- *Chapter 5* lists a few examples of experimental results using the application. This chapter mainly focuses on the analysis of performance of three different CF algorithms in different evaluation metrics given different datasets.
- *Chapter 6* concludes the project and gives directions for future work.

Chapter 2: **Background Research**

2.1 Recommender Systems

Information workload has become an increasingly important issue these days [3]. No matter what information items a person is interested in knowing more about, there are some recommender systems (RSs) online that recommend relevant items [11]. Recommender systems have shown great potential to help users find interesting and relevant items within a large information space [9]. Recommender systems have developed into an indispensable part of information filtering system these days. Compared to many other fields of information filtering system, RS is a relatively new discipline which emerged in the 1990s. Since that, the interest in the RS area has increased significantly. It can be said that the popularity of RSs is the result of the information needs in this era. There are several points to explain why recommender systems are so popular and powerful.

The importance of applying recommenders in the modern world is argued by Anderson in his well-known article - the Long Tail [1]. In the article, the author starts with a question asking why the blockbusters or popular products with a high demand are dominating the market and how this type of market has a negative effect on the economy. Then he proposes the Long Tail Theory - a new marketplace mode emphasising the economic benefits from a large number of niches in the tail. In other words, the attention should be paid to the tail where great profits can be gained or even exceed current bestsellers by selling less popular ones but in large quantities. To guarantee a world of abundance and avoid the popularity's monopoly on profitability, what he really urges companies to do is that first the online streaming stores which support a wider distribution channel of products should be put into practice and then great long tail businesses should use lower prices to attract customers down the tail. Based on the theory, he finally points out the role that recommender systems play. He describes that recommenders are extremely helpful in driving users to find the unnoticed niche items in the tail. As he said, "a good recommendation can ease customers' exploration of the unknown as well as end the tyranny of the hit, consequently manifesting the power of the long tail".

Anderson's theory helps to inspire more service providers to rely on recommenders to increase their turnovers. Besides, there are extra points indicating the power of recommender systems. First, recommender systems lead to a profitable gain for some brick-and-mortar stores [25]. Second, good recommender systems improve the effectiveness of exploring useful items because RSs can suggest different items to different users according to their preferences. For instance, the well-developed recommendation engine employed on the video-stream online website YouTube [52], recommends videos to users by analysing user historically-recorded data. With the aid of the recommender, users are able to find possibly preferred items more easily. Third, RSs play an important role in e-commerce businesses like Amazon where products are recommended to users in a personalised or non-personalised way so as to stimulate users to purchase more products [11].

2.2 Recommender System Dimensions

In literature, there are several aspects to consider a recommender system. Blow are shown key dimensions for describing recommender systems. Combined with these dimensions, some factors associated with evaluating recommender systems are mentioned as well.

2.2.1 Domain

The domain describes the characteristics of content recommended [33]. The domain of recommender systems can be, such as a top-10 of news or article recommendations in your inbox, videos recommended on a digital streaming website, products on some e-commercial platforms, accommodations from a house renting website, or even honeymoon destinations for a recently-married couple [34], etc. It is important to take into account the domain when designing a recommender system. This is because, first, it determines the severity of erroneous recommendations for a recommender system [34]. In another words, the consequence of relevant product recommendations in Amazon is possibly not as severe as that of medicine recommendations to patients. Second, as proposed by [33], different recommendation domains provide different opportunities for the gathering and application of knowledge, which lead to a mapping between domain characteristics and recommendation technologies. Last, it reflects whether it is appropriate for a RS to make repetitive recommendations. For example, in the domain of music, the songs that have been recommended to users are not appropriate to recommend again.

Based on the functions that the domain has, this dimension acts as a vital guideline for choosing metrics in evaluation experiments. To be specific, the medicine recommendation example requires evaluators to lay more stress on the accuracy metric since the cost of inaccurate recommendations in that domain is high. Whereas the movie suggestion example asks evaluators to measure if the recommender system is capable of suggesting various, novel and diverse movies, namely, placing more emphasis on the beyond accuracy metrics.

2.2.2 Context

The context focus on the environment in which recommendations are generated [7, 11]. A concept related to the context is the contextual information which refers to the background information of the recommendation receivers, including the time, mood, location and weather they are in, or the people that are round them, etc. The contextual information can very important in some domains and improve the performance of recommender systems, such as the time and location information for nearby restaurants recommendation. Due to the importance, the context-aware recommender systems as a subclass of recommendation algorithms has been well performed to improve the quality of recommender systems. A typical example is given in [35], which describes three different algorithmic paradigms for incorporating contextual information into the recommendation process.

The direct way to test the difference between the contextual information considered and ignored is to conduct an online A/B trial [26]. The strategy is to evaluate the different cases by analysing real usage and recommendation feedback. The feature of interaction with users makes online testing possible to evaluate a recommender system with contextual information injected. Since the contextual information is closely related to user environment, changes at different times, and has to be collected in real-time and instantly, it is difficult for offline experiments to evaluate the performance of context-aware recommender systems.

2.2.3 Purpose

The purpose is another important dimension to describe recommender systems. The purpose can be defined as the goals for which a recommender system is being used [33]. The purpose can be the further study in recommendation algorithms for researchers, or online testing for evaluators, or simply the retrieval of relevant information for internet users. Understanding for the purpose is often important in real marketing because the purpose of users and information providers can sometimes be different [34]. For instance, the users may be more interested in receiving high-quality recommendations, while the providers are more willing to suggest particular items or products to increase the revenue. That says the purpose guides the recommender employers to be on the right track.

In [7], it is emphasized that, in order to properly evaluate a recommender system, it is important to understand the user purpose. The paper points out that a comprehensive consideration for user goals makes it more sensible to decide what evaluation metrics are chosen to measure the quality of recommender systems.

2.2.4 Personalisation Level

Recommendations can be made at major two levels of personalisation, personalised and non-personalised.

- **Non-personalised.** Non-personalised recommender systems recommend popular and relevant items to users. They are often found on a news site, where a list of top-n popular news is generated to users, or on an e-commercial website where mutually-associated products are suggested to consumers, such as the product association recommendation on Amazon [21]. Non-personalised recommender systems are able to recommend or predict items that users are more interested in than randomly selected items [11]. One advantage of non-personalised recommender systems is that the data model for forming recommendations is easier compared to personalised recommender systems.
- **Personalised.** Although non-personalised are useful in certain areas, the personalised recommendations are most studied in the area [11]. The well-known non-personalised recommendation example is Amazon's Recommended For You list [53], where recommendations are generated specifically for the target consumer. In another words, personalised recommendations are customized based on user preferences or profiles. It is proved that personalised recommendation is of great significance for the user's decision-marking processes [7], which means the user's behaviours such as which item to check first, what products to buy and which items to go through in detail can be greatly influenced by personalised recommenders. Besides, a learning process is likely to help optimise and advance the personalised recommenders [42]. It is reasonably acknowledged that all kinds of information or data about users collected by a personalised recommender can be used to improve the next round of recommendations. In other words, the constant feedback from user-system interactions can help the evolution of recommender systems.

2.2.5 Input and Output

Input and output of a recommender system describes the interface on which the recommender system generates recommendations [34]. Input refers to the ratings or taste preferences that users add into the system. For example, Netflix once enabled users to enter their preferences on genres and topics [34]. The preference information is said an explicit input if the system has a user state his opinion about a given item [2, 14]. An example about the other kind of input is the video recommendations in Youtube, which are made by analysing user behaviours that were produced in previous interactions with the system, clicking of a specific video for example. This input by which the system tries to infer user taste is described as implicit input [2, 14]. The output features the results that recommender systems generate. Output can be presented to users in many forms. It can be an item prediction, a list of non-personalised recommendations, or a top-10 list of personalised recommendations. In literature, the output is divided into two categories. Recommenders with the ability of explaining recommendations are called white-box recommenders, otherwise black-box recommenders [2]. An obvious explanatory recommender system is the one on Amazon, which suggests a product to users with explaining because the users bought another product similar to this one. It was found [54, 2] that explanatory recommender systems can convince users to buy and gain users' trust in the system as a whole.

2.3 Taxonomy of Recommendation Algorithms

There are so many well-established recommendation algorithms. In literature, they can be generally divided into the following categories.

- **Content-based.** Algorithms that use content metadata and user profiles to calculate recommendations are called content-based algorithms [34]. It works the way that new interesting items are recommended to a user by matching up the attributes of the user's profile with the attributes of a content item [12]. In order to achieve a content-based recommender system, the most important task is to calculate the item-item similarity or map items to profiles. Often, the content representation plays a role in the computation, namely extracting important item contents and representing them into data structures. The traditional content-based algorithms use information retrieval techniques to represent items in an unstructured manner (using the raw contents of documents). However, for items that are developed by categorical features, such as commodities on e-commercial sites, an alternative content-based approach named case-based uses a well-defined set of features and feature values to represent items in a more structured manner, allowing for fine-grained judgements about the similarity between items [37].

- **Collaborative filtering (CF).** As Jennifer Pahlka said "When one neighbour helps another, we strengthen our communities.", this quote exactly gives a general picture of how a CF system works. Further speaking, a CF system attempts to generate recommendations by combining the preferences of similar users in the system [3, 2, 7, 4, 5, 11]. Its characteristic of collaborative filtering makes it perform well in personalised recommender systems [2]. As stressed by Herlocker, CF systems are based on human rather than machine analysis of content, which causes them having many significant advantages over traditional content-based techniques [2]. The advantages are concluded in literature: (a) the ability to filter various content, (b) the ability to filter items based on taste and quality, (c) the ability to make serendipitous recommendations. Nevertheless, some unavoidable limitations are also proposed by some researchers. For example, Herlocker et al. [2] point out that the stochastic computation processes and data sparsity in a CF system can lead to being not trusted for high-risk content domains, such as medicine recommendation. This CF techniques are what the project specifically falls into. Hence, more technical details of the three CF algorithms, user-based, item-based and matrix-factorisation in this category are revealed in Chapter 3.

- **Hybrid.** As known, content-based techniques make recommendations built upon content models while collaborative filtering techniques have good performance in recommending novel and serendipitous items [3]. In order to share the advantages of individual algorithms, hybrid approaches involving two or more recommendation algorithms has been proposed [16, 28]. An example of implementing a hybrid system by combining CF and content-based approaches can be found in [28].

Apart from the algorithms listed, there are many other different algorithms that have sprung up recent years. For example context-aware recommender systems make recommendations by taking contextual information into account [35]. Constraint-based [27] approaches are used to make the product selection process more effective in complex products such as financial services or electronic consumer goods. Conversational approaches are designed to provide an interactive process between the system and the user for collecting user feedback information [30]. Due to this feature, conversational algorithms are capable of overcoming the problem of insufficient understanding for user preferences in many other recommender algorithms that use the user input data only once to make recommendations [11]. Demographic approaches user attributes classified as demographic data, such as gender, race, age, etc., to make recommendations [38]. An example proving that demographic data can be used to enhance the CF algorithms can found in [38].

2.4 Evaluating Recommender Systems

In the section, a review of several methods and metrics to evaluate collaborative filtering recommender systems is stated.

2.4.1 Evaluation Methodology

the efficiency of online evaluation will be low if a large number of recommender algorithms are chosen to be conducted. Hence, an initial filtering for so many existing recommender algorithms becomes necessary. This is how another methodology offline evaluation [7] is introduced. Offline evaluation works to evaluate core components of recommendation algorithms using existing user-rating datasets. There are several techniques [32] to conduct offline evaluation experiments. They are leave ratio out(LRO), k-fold cross validation(KCV) and leave-one-out(LOV). LRO randomly partitions the data into training and test set with a ratio value indicating the percentage of train set. KCV randomly splits the dat set into k samples. For each sample, one of the k samples as the test set is evaluated based on predictions or recommendations made using the remaining k-1 samples as training set, repeating k times to guarantee every subsample has been used as the test set. LOV selects a single user-item-rating as the test data with the remaining as the training set, repeating for each of the observations to guarantee every observation has been used as the test set. The result of LRO is given by the measurement score in the test set and the final results of KCV and LOV are gained by averaging over single round of evaluation.

2.4.2 Accuracy Metrics

In terms of performance evaluation metrics, the most frequently used metrics goes to accuracy metrics. There are several reasons why accuracy metrics are normally considered in the first place. First, accuracy is an essential performance indicator telling us how close the received items are to users, namely the relevance with the user needs. Second, accuracy metrics act as important reference to beyond accuracy metrics that are discussed in next section. According to the end-user tasks, Herlocker et al. [7] propose that recommender systems work with various tasks, among which two most common tasks are generating a prediction and recommending a list of items. The accuracy metrics can be broadly divided into two categories. The prediction metrics are suitable for the prediction task, while top-N metrics are better for the recommendation task.

2.4.2.1 Prediction Accuracy

Mean Absolute Error: In offline evaluation for the prediction task, normally the dataset is split into train set and test set. The train set is used to train the prediction model and items in the test set are applied for the performance evaluation as actual ratings for the predicted items are known. Intuitively, the direct way to measure the accuracy of the items predicted by a recommender system is taking the difference between the predicted and actual ratings. This is called mean absolute error (MAE) [10, 7, 17]. It is calculated as Equation 2.1.

$$MAE = \frac{\sum_N |p_{a,j} - r_{a,j}|}{N} \tag{2.1}$$

Where $p_{a,j}$ is the predicted rating of user a for item j, $r_{a,j}$ is the actual rating of user a for item j and N is the number of all user-pair items in the test set.

Mean Square Error & Root Mean Square Error: Although it is easy to implement and understand, MAE does not distinguish the difference between large errors and small error, i.e, large errors are not penalised. In order to deal with the problem, two alternatives are introduced. They are mean squared error (MSE) [10, 2] calculated as Equation 2.2 and root mean square error (RMSE) [10, 2] calculated as Equation 2.3. In the two approaches, the error is squared to penalise large errors.

$$MSE = \frac{\sum_N |p_{a,j} - r_{a,j}|^2}{N} \tag{2.2}$$

$$RMSE = \sqrt{\frac{\sum_N |p_{a,j} - r_{a,j}|^2}{N}} \tag{2.3}$$

2.4.2.2 Top-N Accuracy

When conducting the recommendation task, a list of recommendations are generated to the active user. Top-N accuracy metrics are required for the task. As described by [7], top-N accuracy metrics are also called classification accuracy metrics(CAE) used to evaluate the top-N recommendation relevance. The top-N metrics ask how correct or incorrect the decisions are made by a recommender system [2]. CAEs are tolerant to the deviation between predicted ratings and actual ratings, but are strict to the performance of how well the recommender system classify items, namely, the relevance, positions, orderings of items in the top-N lists. Precision and recall from the field of information retrieval are two classification metrics frequently applied to measure the relevance of top-N recommendations.

Precision & Recall: Precision [7, 11, 34] represents the percentage of relevant items in the top N recommended items. It is described in formula as Equation 2.4. Recall [7, 11, 34] represents the percentage of relevant items that are recommended. It is described in formula as Equation 2.5.

$$Precision = \frac{|REL \cap REC|}{|REC|} \tag{2.4}$$

$$Recall = \frac{|REL \cap REC|}{|REL|} \tag{2.5}$$

where REC and REL represent the recommended and relevant items in the test set respectively.

F-Measure: Since any one of precision and recall can be manipulated without reference to another one. For example, with the increasing number of items recommended, recall rises and precision declines. F-measure [43, 21] is introduced to combine the two metrics to provide a single score indicating the relevance of recommended items. F1 [21] is one of the F-measure implementations, calculated as Equation 2.6.

$$F1 = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{2.6}$$

Other approaches: Due to certain limitations in precision and recall [7], many other top-N metrics are proposed as alternatives. For example, mean accuracy precision (MAP) computes the average precision across several different levels of recall [2]. Also, another approach named receiver operating characteristic (ROC) [20] provides a theoretically grounded alternative to solve the problem of only considering binary relevance in precision and recall. Besides, normalised discounted cumulative gain (nDCG) [22] places emphasis on the positions of recommended items, where relevant items at lower ranks are penalised.

2.4.3 Beyond Accuracy Metrics

Prediction accuracy metrics measure how close the items predicted differs from the actual ratings and top-N metrics measure how relevant the items are recommended [7], while beyond accuracy metrics shift focus towards other properties which significantly describe the performance of recommender systems [8]. Those properties include popularity, coverage, diversity, novelty, etc. The measurements of beyond accuracy metrics are considered difficult because they often involve in the levels associated with user subjective opinions such as user emotions or biases [16]. Despite the difficulty, in literature, still many approaches have been proposed to measure them. In this section, the beyond accuracy metrics involved in the project are reviewed first and then some other out-of-accuracy considerations are mentioned as well.

2.4.3.1 Popularity

In order to better understand the performance of recommender systems, popularity has to be considered. Popularity as an important metric describes how popular the recommended items are. This implies that it is particularly important for e-commerce businesses. As described by [1], if a recommender system is only able to generate popular items, that means there is only around 20% item discovery in the catalogue space leaving the remanding 80% niche items unexplored. In real offline evaluation, a recommender system with lower popularity score is considered well-performed in exploring potential unpopular items. One simple way to measure the popularity of items recommended to a user is averaging over the percentage of each item's popularity. The equation below shows that the popularity of item j is calculated as the number of users who rated item j (N_j) divided by the number of users in the system (N).

$$Popularity(j) = \frac{N_j}{N} \tag{2.7}$$

2.4.3.2 Coverage

Coverage refers to a range of products for which a recommender can make a prediction or recommendations [2]. A recommender with a high converge will make a big difference to users. Many papers have considered it as one of the most important metrics to measure a recommender system [42, 7, 8]. In literature, the coverage is often described from three perspectives. First, it can be measured simply by the percentage of target users for who at least one recommendation can be made. This type of coverage tells if all users are likely to receive recommendations in the system. Second, it can be measured by asking "what is the percentage of items that the system is capable of making a prediction or recommendation". This is so called prediction coverage or recommendation coverage [7, 8]. This coverage indicates how large the recommendation space is out of the item space. A simple way to calculate this type of coverage is taking the percentage of items in the dataset which appear at least once in the top-N recommendations made over all target users [8]. The measurement formula is given as follows.

$$RecCoverage = \frac{|I_p|}{|I|} \tag{2.8}$$

where I is the set of available items and I_p refers to the set of items that are recommended at least once to users.

In another aspect, coverage can be measured by asking "out of the available items, how many items are ever recommended to users". This is so called catalogue or item-space coverage [7, 8, 22]. This type of coverage is an important consideration for the performance of recommender systems because a recommender system that are able to make a large portion of item space in its recommendation list is deemed high-quality [8]. A simple way to calculate this type of coverage is averaging the percentage of items out of the available items that are in a given target user's candidate recommendation list across all target users. It is calculated as Equation 2.9.

$$ItemCoverage = \frac{\sum_{\forall u \in U} \frac{C_u}{|I|}}{|U|} \tag{2.9}$$

where I and U is the set of available items and users respectively, and $|C_u|$ refers to the number of candidate items for user u.

There is a bottleneck in the method described above. It does not take individual item into account and ignore the number of each item's occurrence in the candidate recommendations. To alleviate the problem, an optimised approach called Shannon entropy is proposed. As described in [22], the Shannon entropy is calculated as Equation 2.10.

$$EntropyCoverage = -\sum_{i=1}^{L} p_i \times log(p_i) \tag{2.10}$$

where p_i is the percentage of item i that is in the recommendation lists and L is the number of candidate items.

2.4.3.3 Diversity

As opposed to similarity, diversity is defined as a metric frequently applied to the top-N list of recommendations to measure how different the items recommended are [14, 44, 7]. Recommender systems with diverse items recommended can avoid users to explore repeated and useless items, which is not appropriate for the user task in finding all good items [7]. In relevant studies, the most commonly used approach to measure diversity is pairwise dissimilarity [14], where the similarities between all pairs of items in the top-N recommendation list are computed. Equation 2.11 shows how it is calculated.

$$Diversity(r_1, ..., r_N) = \frac{\sum_{i=1...N} \sum_{j=1...N \wedge i \neq j} (1 - sim(r_i, r_j)))}{N(N-1)} \times 2 \tag{2.11}$$

where the diversity is calculated as a function of the top-N recommendations $r_1, r_2, ..., r_N$ and $Sim(r_i, r_j)$ is the similarity between item i and j which can be computed by similarity functions such as a content-based similarity function, the cosine similarity or Pearson correlation described in Section 3.1.

2.4.3.4 Novelty

Another important consideration into beyond accuracy metrics is novelty. It measures the ability of a system to suggest items that users are not aware of [44, 3, 45, 8]. It is an important performance indicator because recommender systems with a lack of novel recommendations make users less know the existence of some items in the catalog [22]. Novelty is often closely related to popularity, or to say opposite to popularity. For example, an approach to measure novelty is called self-information-based novelty [46] which assumes that popular items provide less novelty. It is calculated as follows.

$$S(N) = \frac{\sum_{U \in U_{TestSet}}(-\sum_{i=1}^{N} log(\frac{|U|}{rels_i}))}{|U_{TestSet}|)} \tag{2.12}$$

where $U_{TestSet}$ is the set of users in the test set and $|U|$ is the number of test users, and $rels_i$ represents the number of users who have rated item i.

Concerning novelty, it is highlighted that recommendations will make no sense if they are just novel but non-relevant to the user [7]. Hence, along with other beyond accuracy metrics, novelty should be measured in combination with accuracy metrics.

2.4.3.5 Other Beyond Accuracy

In literature, there are many other metrics proposed to describe beyond accuracy properties of recommender systems. Serendipity measures the ability of recommending items that are surprising and attractive to users [13, 8, 7]. Compared to novelty, the difference between them is that serendipitous items are those that are less likely to be found by the users, whereas novel items usually are those that users can find but just are not aware of. Confidence is a metric measuring how sure is the recommender system that its recommendation is accurate [7]; the learning rate measures how quickly an algorithm can produce good recommendations [7]; privacy is used to measure the extent to which recommenders use privacy protected data and proven to a user that data is safe [23], etc.

2.5 Conclusion

This chapter first included the introduction to different dimensions of recommender systems and in these dimensions some considerations important for evaluating recommender systems were mentioned. Then a brief description of the taxonomy of recommendation algorithms was given. Last, the core background research went to the evaluation of recommender systems, which presented two main types of evaluation metrics, accuracy metrics and beyond accuracy metrics. The algorithms listed in this chapter will particularly serve as reference to the implementation and experiment analysis of the project which will be discussed in Chapter 4 and 5. In the next chapter, the focus will fall into the three collaborative filtering algorithms implemented in the project.

Chapter 3: **Collaborative Filtering Algorithms**

According to the taxonomy of recommendation algorithms, there are main two subclasses of collaborative filtering(CF) algorithms, neighbourhood-based and model-based. Two neighbourhood-based approaches are user-based [17] CF and item-based [4] CF. Both approaches try to find nearest items or users that are similar to the active user or item. Model-based approaches involves extracting important information from datasets and use the information as a model to make recommendations. One prominent approach in this family is matrix-factorisation [5, 39]. Here the following gives the specific implementations of the three CF algorithms.

3.1 User-based

User-based [17] approaches form recommendations or make predictions by finding users whose tastes are similar to the active user. In order to implement a user-based CF algorithm, the following steps need to be considered.

3.1.1 Data Representation

Before calculating the similarities between users, the preference data of users for items has to be represented. Here for simplicity and good illustration, the preference data is given by explicit ratings on a scale of 0 to 5. Table 3.1 shows an example of data representation where users' ratings for items are represented in the form of matrix. There are five rows and four columns in the matrix. Each row refers to a user labeled by u_1 ... u_5 and each column stands for an item labeled by i_1 ... i_4. Here N and M are defined to represent the number of users and items respectively. The number in row i and column j represents the rating of user i for item j. For example, the rating of u_5 for i_2 is 2. Besides, the blank cells of the table indicate that the users have not rated the items.

	i_1	i_2	i_3	i_4
u_1	3		3	4
u_2	4	1	4	2
u_3	5	2		2
u_4	2	5	3	
u_5	1	2	4	1

Table 3.1: An example of user-item matrix in data representation

3.1.2 Similarity Computation

With the user preference data has been collected, the next step is to compute the similarity between the active user and all other users in the system. This is an important step for building the relationship model between users in user-based algorithms.

There are many methods proposed in literature for the similarity computation, such as, mean square difference [18], and some extensions like significance weighting [3], Inverse User Frequency (IUF) [10], etc. Here are given two popular methods used to compute the similarities between users.

The equation below gives the computation of the vector/cosine similarity [10]. In this approach, users are represented as vectors in the M-dimensional item space and the similarity between user a and i is calculated as the cosine of the angle between the vector of user a and the vector of user i. This approach will generate a result between 0 and 1 if $r_{i,j} >= 0$, otherwise between -1 and 1.

$$w_{a,i} = \frac{\sum_{j \in I_a \cap I_i} r_{a,j} r_{i,j}}{\sqrt{\sum_{k \in I_a} r_{a,k}^2 \sum_{k \in I_i} r_{i,k}^2}} \tag{3.1}$$

where I_a and I_i is the set of ratings user a and j for all items respectively and $r_{a,j}$ refers to the rating of user a for item j.

An alternative approach to calculating the similarity is Pearson correlation [15]. The following equation defines the similarity calculation between user a and i. In this approach, the summations are calculated over co-rated items. The approach will generate results in a value of -1 to 1. 1 indicates total agreement on co-rated items, 0 implies no similarity between users and -1 indicates total disagreement.

$$w_{a,i} = \frac{\sum_{j \in I_a \cap I_i} (r_{a,j} - \overline{r}_a)(r_{i,j} - \overline{r}_i)}{\sqrt{\sum_{j \in I_a \cap I_i} (r_{a,j} - \overline{r}_a)^2 \sum_{j \in I_a \cap I_i} (r_{i,j} - \overline{r}_i)^2}} \tag{3.2}$$

where $\overline{r}_a$ refers to the mean of all ratings given by user a.

After the similarity computation, each user builds a collection of similarities to the rest of users.

3.1.3 Neighbourhood Formation

Once the similarities are calculated, the next step goes to neighbourhood formation. In another words, the top most similar users needs to be selected as the active user's neighbourhood. There are different approaches to form the neighbourhood. One commonly-used way is k nearest-neighbour (KNN) [55], which forms a neighbourhood of size k by selecting the k users with the highest similarity to the active user. The neighbourhood size k is typically chosen by experiment. It has to be picked carefully because accuracy will be reduced (the influence of the most similar neighbours is diluted) if a large k is picked, while accuracy will be poor (users do not have enough close neighbours) if a small k is picked.

3.1.4 Make a prediction or generate a top-N list

After the formation of neighbourhood, the predicted rating of an item for the active user is calculated by taking a normalised weighted average over the ratings for the item that are from each of the user's neighbours. One commonly-applied approach of calculating the predicted ratings is Resnisk's formula [15, 3]. Equation 3.3 gives the computation of predicted rating of user a for item j. This approach

$$p_{a,j} = \overline{r}_a + \frac{\sum_{i=1}^{k} w_{a,i}(r_{i,j} - \overline{r}_i)}{\sum_{i=1}^{k} |w_{a,i}|} \tag{3.3}$$

where $w_{a,i}$ is the similarity score between user a and i and k is the size of neighbourhood.

Concerning the task of generating a top-N list, one way is described as follows: First a candidate set of items for recommendation needs to be formed by taking the union of all the items that have been rated by the active user's neighbours. Second, the items that have already been rated by the active user are filtered out from the set. Third, the remaining items ranked by the predicted ratings calculated as Equation 3.3 for example and finally the top-N ranked items are returned as recommendations to the target user.

3.2 Item-based

There is a bottleneck in user-based CF algorithms. User-based CF algorithms suffer from serious scalability problems [23]. The problem depicts that the computations are expensive in real-world systems where there are potentially millions of users and items. This is how item-based algorithms are introduced to overcome the bottleneck by finding relationships between items rather than users. Item-based CF algorithms [4] are based on the intuition that users are interested in items similar to those previously experienced.

The item-based implementation goes roughly the same process as the user-based one, namely, the four steps introduced in Section 3.1. The step of data representation is the same. In similarity computation, the pairwise similarities are calculated based on items instead of users. For example, in the cosine similarity approach, items are represented as vectors in the N-dimentional user space and the similarity between item a and j is calculated as the cosine of the angle between the vector of item a and the vector of item j. As a result, after this step, there will be a data structure produced to store the similarities between items rather than users. In neighbourhood formation, an item neighbourhood is formed by taking a subset of other items with the highest similarity to the active item. In making a prediction, taking the Resnisk's algorithm as an example, the prediction equation 3.3 is changed to

$$p_{a,j} = \overline{v}_j + \frac{\sum_{i=1}^{k} s_{i,j}(r_{a,i} - \overline{v}_i)}{\sum_{i=1}^{k} |s_{i,j}|} \tag{3.4}$$

where $s_{i,j}$ is the similarity score between item i and j, $\overline{v}_j$ refers to the mean rating of item j across all users, $r_{a,i}$ refers to the rating of user a for item i and k is the size of neighbourhood.

The process of generating a top-N list of recommendation works the same way as the user-based as described in Section 3.1.4.

3.3 Matrix Factorisation

There are some limitations in user-based and item-based CF algorithms. First, neighbourhood-based algorithms heavily depend on the relationships between users and items in order to form predictions or recommendations. This indicates that a limited quantities of data cause it hard to build the relationship model (the similarities between neighbours). This is so called the data sparsity problem [50]. Second, they suffer from the early rater problem [51]. This issues describes that predications are hard made for items that are newly added to a system as the items are rarely rated in the system (not enough ratings to calculate predictions). It also depicts that predictions or recommendations are hard made for users who newly registered as they do not show enough their taste in the system (not neighbours).

Due to these limitations, the model-based approaches are introduced. There are many well-established model-based approaches, among which the matrix factorisation [5, 39] approach is widely-used. It works by making an assumption that there exist some latent features which explain user preferences. Its goal is to uncover these latent features which can afterward be used to predict items for a user more accurately [29].

It starts with initializing the number of features K and two matrices P ($|U| \times K$ matrix) and Q ($|I| \times K$ matrix), which represents the strength of the association association between a user (P) or an item (Q) and each of the K features. Then the multiplication of P and Q^T is calculated as the prediction matrix in which the predicted ratings are presented. The following graph gives a nice illustration of how the matrics are constructed.

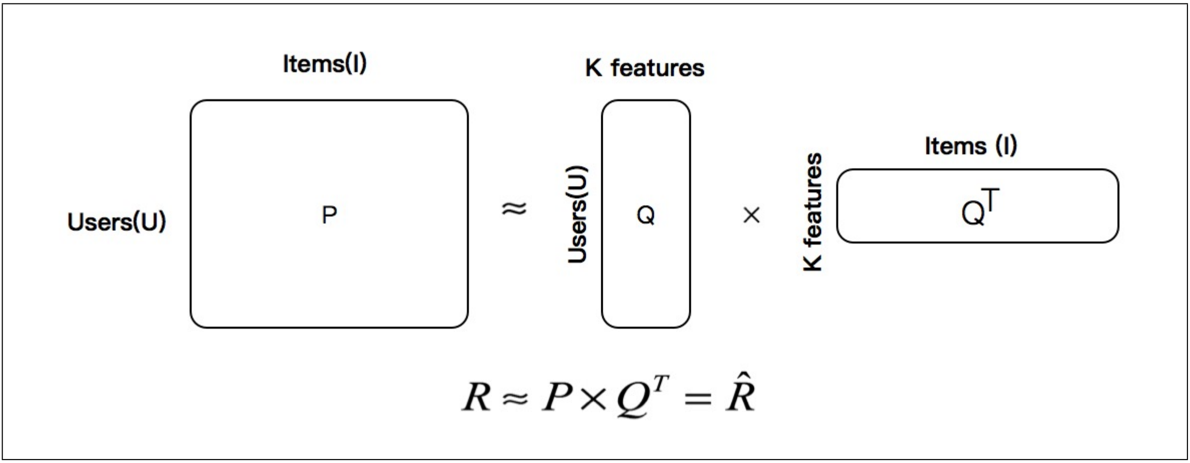


Figure 3.1: Illustration of matrix factorisation whose goal is to find the predicted rating matrix $\hat{R}$ by learning user and item feature values

With P and Q calculated, a predicted rating for user i, item j is given by the dot product of the two vectors corresponding to user i and item j, namely, $\hat{r}_{i,j} = p_i^T q_j = \sum_{k=1}^{K} p_{ik} q_{kj}$. In order to approximate the actual user-item matrix, iterations are assigned to update P and Q and the difference, namely the loss function is given by the error between the predicted and actual ratings. There are many ways to compute the error. One example is square error loss [18]. The following equation specifies the square error over all ratings in the training set T.

$$E = \sum_{r_{i,j} \in T} e_{i,j}^2 = \sum_{r_{i,j} \in T} (r_{i,j} - \sum_{k=1}^{K} p_{ik} q_{kj})^2 \tag{3.5}$$

After obtaining the loss function, a decision needs to be made upon how to modify P or Q so that the error can be minimised. Many approaches have been proposed to achieve this goal, for example, the stochastic gradient descend (SGD) [40] introduced by Simon Funk[1]. It works by updating the values of p_{ik} and q_{kj} according to the direction of the gradient. For each factor, it updates the values of p_{ik} and q_{kj} each time using update rules by looping through all each rating

[1]http://sifter.org/simon/

in the training set T until the error is minimised. Equation 3.6 gives the update rules for both p_{ik} and q_{kj}.

$$\begin{aligned} p_{ik}^{'} &= p_{ik} + \alpha \frac{\delta}{\delta p_{ik}} e_{i,j}^2 = p_{ik} + 2\alpha e_{i,j} q_{kj} \\ q_{kj}^{'} &= q_{kj} + \alpha \frac{\delta}{\delta q_{kj}} e_{i,j}^2 = q_{kj} + 2\alpha e_{i,j} p_{ik} \end{aligned} \tag{3.6}$$

where α is the learning rate normally given by 0.001.

With the loss function and update rules (SGD), the difference between the prediction matrix and the actual user-item matrix is minimised. The above describes a basic process of the implementation of matrix factorisation techniques. In literature, there are many variations to implement matrix factorisation algorithms. For example, the approach called linear matrix factorisation(LMF) proposed by [19, 5] works by adding two bias constants and regularisation parameters to the prediction model so that the overfitting problem and latent user biases can be avoided. In this approach, the predicted rating of user i for item j is changed to

$$\hat{r}_{i,j} = p_i^T q_j = \sum_{k=1}^{k} p_{ik} q_{kj} + c_i + d_j \tag{3.7}$$

where c_i and d_j are two constants representing the bias of user i and item j respectively.

It is noteworthy that the two constants need to be trained together with p_{ik} and q_{kj}. The update rules for c_i and d_j are specified in Equation 3.8.

$$\begin{aligned} c_i^{'} &= c_i + \alpha(\frac{\delta}{\delta c_i} e_{i,j}^2 - \sigma c_i) = c_i + \alpha(2e_{i,j} - \sigma c_i) \\ d_j^{'} &= d_j + \alpha(\frac{\delta}{\delta d_j} e_{i,j}^2 - \sigma d_j) = d_j + \alpha(2e_{i,j} - \sigma d_j) \end{aligned} \tag{3.8}$$

where σ is the regularisation parameter for alleviating the overfitting problem normally given by 0.01.

With the bias constants c_i and d_j and the regularisation parameter added, now the error function becomes

$$E = \sum_{r_{i,j} \in T} e_{i,j}^2 = \sum_{r_{i,j} \in T} ((r_{i,j} - \sum_{k=1}^{K} p_{ik} q_{kj} + c_i + d_j)^2 + \frac{\beta}{2}(\|p\|^2 + \|q\|^2) + \frac{\sigma}{2}(c_i^2 + d_j^2)) \tag{3.9}$$

where β is an additional regularisation factor used to suppress overtraining and therefore improve performance on unseen examples [5]. They are normally given by 0.01.

With the loss function and update rule, the same process as introduced earlier in the section is applied to train the predicted matrix $\hat{R}$ as closest as possible to the actual prediction model by learning the user-item matrix R.

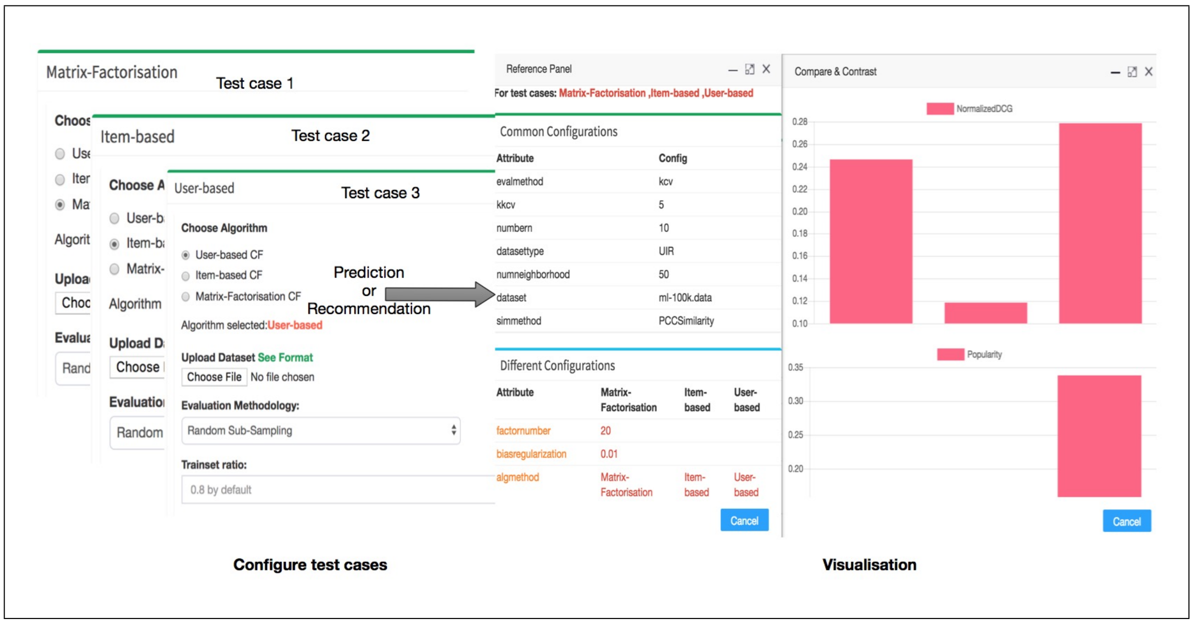
The LMF algorithm as a variation of matrix factorisations is implemented in this project, but renamed to biased matrix factorisation(BMF). More details about its performance is discussed in Chapter 5.

Chapter 4: **Architecture and Implementation**

With the three collaborative algorithms detailed, this chapter will first give a high-level overview and architecture of the system, and then presents the implementation from two aspects, front-end and back-end respectively, and finally reveals the methodology for testing the system.

4.1 System Overview

The following gives the overview of the system, illustrating how a comparative evaluation experiment is conducted in the web application



Choose Algorithm

- User-based CF [✓]
- Item-based CF
- Matrix-Factorisation CF

Algorithm selected:User-based

Upload Dataset See Format

Choose File No file chosen

Evaluation Methodology: Random Sub-Sampling

Trainset ratio: 0.8 by default

For test cases: Matrix-Factorisation ,Item-based ,User-based

Common Configurations

Attribute	Config
evalmethod	kcv
kkcv	5
numbern	10
datasettype	UIR
numneighborhood	50
dataset	ml-100k.data
simmethod	PCCSimilarity

Different Configurations

Attribute	Matrix-Factorisation	Item-based	User-based
factornumber	20		
biasregularization	0.01		
algmethod	Matrix-Factorisation	Item-based	User-based

Figure 4.1: System overview of the web application

As seen from Figure 4.1, the evaluation experiment is conducted in the from of test case. For a user, he can create, delete, or hide test cases. A user is allowed to conduct multiple test cases. Normally, a test case corresponds to evaluating a recommender system. In order to evaluate a recommender system, some configurations needs to be done first, such as given the recommendation algorithm, selecting evaluation metrics, etc. After the configurations, two options are provided to run the test case. Users can use the configured algorithm to make predictions as well as form a top-N recommendations. With the algorithm run, the corresponding evaluation metrics are gained. These obtained metrics are finally used for visualisation. The same process is applied to other use cases for different recommendation algorithms so as to compare and contrast the performance between algorithms.

4.2 System Architecture

The following figure shows the architecture of the web application, indicating three components making up the system.

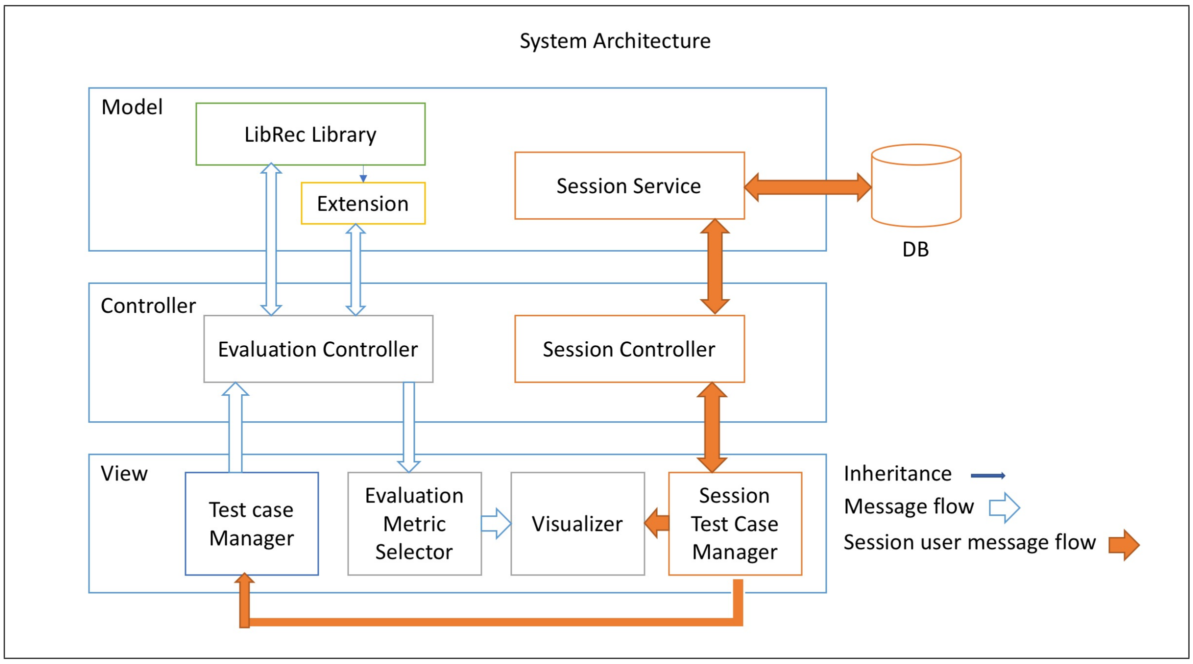


Figure 4.2: System architecture of the web application

4.2.1 Interface

Interface is the web page presentation level responsible for managing the page interpretations to and interactions with users. It is the entrance of users to carry out evaluation experiments via the front-end interfaces. There are several sub modules given to this component, described as follows.

- **The test case manager.** It is used to manage some basic operations of test cases. With the manger, test cases can be conducted in a well-structured, logic and flexible way. Its major message flow goes to the evaluation controller. The message flow is activated when a test case is executed, namely the evaluation request is sent to the recommender systems on the server side. Session Test Case Manager is a little different to the test case manager.
- **Session Test Case Manager.** This manager is a little different to the test case manager. It is used to manage test cases of the host users who are in session services maintained by the server for managing the experimental data generated by the users. There are several connections associated with the module. First, the message flow to the test case manager enables users to initially run test cases as guests. Second, the flow to the session controller is triggered only when users want to pull the experimental results that were stored in previous runs by session service. Third, if the results are pulled, the metrics can be directly used for visualisation through the flow to Visualizer.
- **Evaluation Metric Selector.** This module becomes tangible after the results of evaluation metrics are returned to the front-end. The selector helps users select metrics for visualisation

in an extensible and configurable way. That means metrics can be deleted, added or filtered. The only message flow of this module goes to the visualisation module. The logic flow is reasonable because metrics are first selected and then passed for visualisation.

- **Visualiser.** Visualiser is a very important module in the interface component. Its major responsibility is to visualise different metrics between different test cases via bar charts. One good feature is that there is no restriction on the number of test cases selected for the compare and contrast. Also, there is a functional separation in the module. One functional part is the presentation of test case configurations, which are demonstrated neatly right next to the bar charts as reference.

4.2.2 Controller

Controller acts as the role of handling the requests by users and presenting the results to users. In this project, Two main controller modules are given.

- **Evaluation controller.** Evaluation is the core controller in the controller component. It is primarily in charge of the evaluation tasks and requesting for the service from recommender systems. For example, a dataset file has to be uploaded before running an evaluation experiment. Here the controller calls the uploading service to finish the dataset upload task and the service of recommender system evaluators to complete the evaluating task. This indicates that the controller separates different services for different tasks. The message flow in this module has three places to go. First, the flow to the LibRec[1] library or the extension module allows to call all kinds of services provided by the open-source recommender system library. This flow is almost inevitable whenever an evaluation request is asked. Second, after a recommender system is evaluated, relevant metrics are passed back to the front-end users through the flow to the selector. Last, the flow to the session controller is only available for the case of host users. This is because in that case before metrics are returned, the running data including metrics, log data, running time, etc., has to be passed over to the session controller which helps to store the data to database.
- **Session Controller.** Session controller has two responsibilities. First, it controls the creation, deletion, retrieval of test cases. Second, it controls the storage and retrieval of evaluation metrics. The controller plays a role when the front-end users are detected ones who are using session code. Unlike guest users, host users' operations for test cases or evaluation are processed by the controller before results are presented to users.

4.2.3 Service

The service is the core level of the system, especially the recommender system service module.

- **LibRec library and extension.** This core work of the project goes to this part. The LibRec library provides a wide range of recommendation algorithms and evaluation metrics. However, this abundance is still insufficient to totally support the project's objectives, for example, no support for many beyond accuracy metrics like popularity, coverage, diversity, etc. This introduces the extension submodule, which inherits classes of the library and is used to write customised algorithms. For an evaluation task, the workflow in the modules is

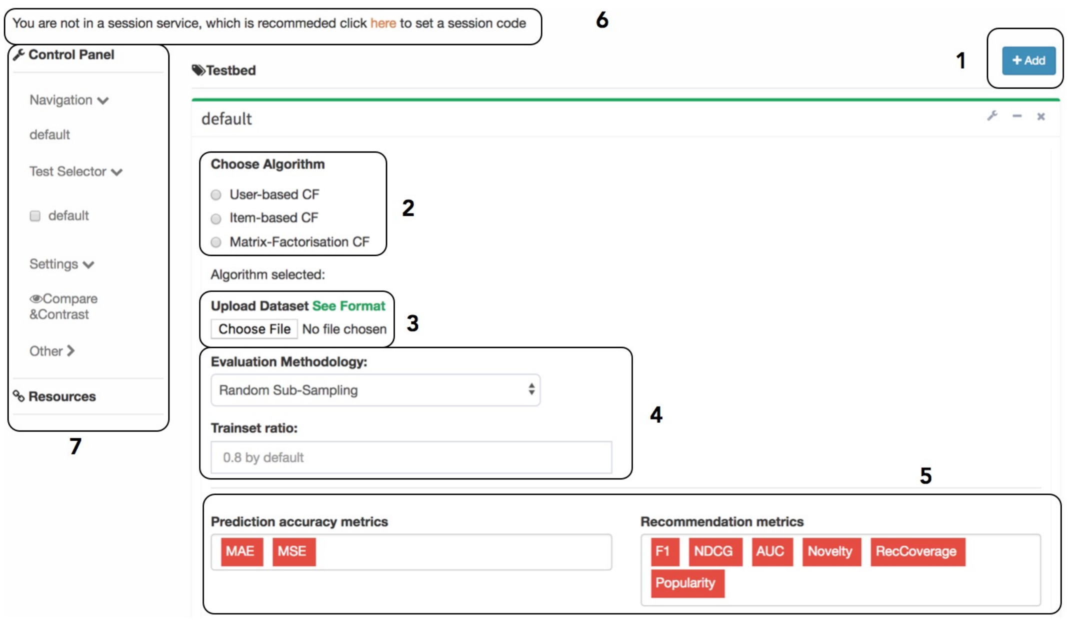
[1] https://www.librec.net/

described as: (1) Map the front-end configurations passed from the evaluation controller to the LibRec standards. (2) Call corresponding recommenders from the library based on the configurations. (3) Feed the prediction or recommendation results to evaluators, directly call the evaluators provided if the configured metrics are found in the library, otherwise the evaluators that are extended in the extension submodule. (4) Return the results to the evaluation controller.

- **Session service.** Session service connects the session controller with database. It describes the back-end functions for the session mechanism. Its main function is the management of permanent data. It directly requests or saves data from or to the database to respond the operations asked by the session controller. By the way, the database is aided by MYSQL[2] and the connection between the service and the database is supported by the MyBatis[3]

4.3 Front-end Implementation

Front-end page programming is an important part of the project because its goal is to build a web application. In the design phase, some goals or to say principles are set up so as to achieve the friendly-interactive application. Figure 4.3 shows a screenshot of the main web page. Blow are the details of good features of the project from the front-end perspective.


You are not in a session service, which is recommeded click here to set a session code

Control Panel

Navigation

default

Test Selector

default

Settings

Compare &Contrast

Other

Resources

Testbed

+ Add

default

Choose Algorithm

User-based CF

Item-based CF

Matrix-Factorisation CF

Algorithm selected:

Upload Dataset See Format

Choose File No file chosen

Evaluation Methodology:

Random Sub-Sampling

Trainset ratio:

0.8 by default

Prediction accuracy metrics

MAE MSE

Recommendation metrics

F1 NDCG AUC Novelty RecCoverage Popularity

Figure 4.3: A screenshot of the application main page

- Responsive. This becomes a good feature because all components in a page are responsible to the size of windows. This makes users easy and comfortable to run the test case in

[2]https://www.mysql.com/
[3]http://www.mybatis.org/mybatis-3/

situations where the application is running on different platforms. The responsive pages are well supported by the CSS framework Bootstrap[4].

- Modularisation. This core of this idea is inspired from Krug in his famous book [47]. He states that a good web application should be as easily and directly operational as possible for users [47]. Based on the theory, the project is modularised at the beginning. As seen in Figure 4.3, several modules are presented, such as, the control panel used for visualising test cases on the left side, and the central panel for experimental configuration, etc.

- Auto adaptive. Due to a wide range of configuration combinations for evaluating recommender systems, a whole configuration presentation on the page will sacrifice the brevity of the application. A good solution to this problem is to make some sections auto adaptive. For example, in Figure 4.3, a hidden section down below the zone 4 including textfields for configurations is presented to users according to the algorithm selected in the zone 2. In order to approach the brevity and convenience to the maximum, many other such examples can be found in the application. The implementation of this feature is mostly done by the front-end scripting language JavaScript and its popular framework JQuery[5] is applied to reduce the heavy work of raw coding.

4.4 Back-end Implementation

The front-end provides users with direct interfaces accessible to the service of recommender systems on the back-end. So to speak, the core work into the research of evaluating recommender systems is done on the server side. The following demonstrates major implementation in the back end.

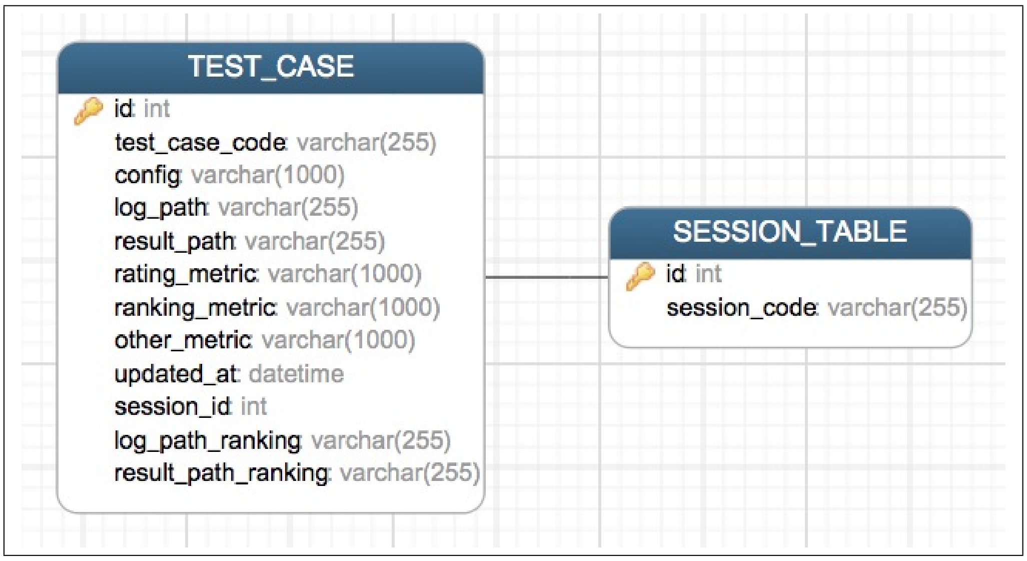

TEST_CASE
id: int
test_case_code: varchar(255)
config: varchar(1000)
log_path: varchar(255)
result_path: varchar(255)
rating_metric: varchar(1000)
ranking_metric: varchar(1000)
other_metric: varchar(1000)
updated_at: datetime
session_id: int
log_path_ranking: varchar(255)
result_path_ranking: varchar(255)
SESSION_TABLE
id: int
session_code: varchar(255)

Figure 4.4: Entity relationship graph of database

- SpringMVC. SpringMVC[6] is a popular model-view-controller framework for building web applications. Its features of clear separation of roles, dependence injection, annotation

[4]https://getbootstrap.com/
[5]https://jquery.com/
[6]https://spring.io/

events enable to build a web application more elegantly and efficiently.

- LibRec. LibRec is the major recommender system open-source library used to support the implementation of evaluating recommender systems in the project. Its features of intensive support for recommendation algorithms and easy extensibility help to enhance and extend the application to a large extent. More details about the library can be found in [31].
- Database Design. Database design highlights the details of session mechanism. Figure 4.4 gives the entity-relationship graph of two tables created in the system. The `test_case` table is used to store important data of a test case and the `session_table` is used to manage session service. The `session_id` in `test_case` table acts as the foreign key reference to the id in session table.

4.5 Testing Methodology

Software testing is an important process in the development lifecycle to verify the correctness, completeness and quality of the application [56]. It aims to reduce the system defects to the minimal before it is released to users. For the project, a strict testing process is involved as well. Based the context of the project, there are three testing scenarios defined to scientifically test the application. Here the point to emphasise is that the application is developed iteratively and incrementally and therefore the testing scenarios below are optimised and conducted along the way.

First, unit testing is also called component testing and is performed on the functional units of the system to check whether they are developed correctly. For example, there defines three unit test classes in the test suite, EvaluatorTest, SessionTest, and ExperimentTest to test different functional units in the system. Second, unlike unit testing, integration testing is a higher level testing scenario where separate modules are combined and tested a group. In another words, it tests the logic flow of integrated functional unitss, as stated in [49]. For instance, in order to test the correctness of conducting an evaluation experiment in a session service, one testing example is made up of the integrated functional units including, create a session service, run a test case, and get evaluation metrics. Last, verification and validation testing is a even higher level testing scenario which is usually carried out at a client location. It focuses on checking whether the system meets the clients' requirements instead of the functional correctness. For this system, the application is developed for recommender system evaluators and the user requirements can be described by the project's objectives listed in Section 1.3. Hence, one considered testing strategy could be to first ask the evaluators to use the application and then ask their feedback on how well the system meets the objectives.

4.6 Conclusion

With main activities in the development process introduced, the next chapter will present a comparative performance analysis of three CF algorithms by conducting several evaluation experiments using the developed application.

Chapter 5: **Analysis of Algorithm Performance**

5.1 Introduction

As highlighted in Chapter 2, evaluating recommender system plays an important role not only in guiding researchers to design better recommender systems, but also in helping service providers to choose between a set of candidate recommendation algorithms. In evaluating recommender systems, in order to avoid one-sided analysis, it is extremely important to measure the performance of a recommender system with a comprehensive selection of evaluation metrics. In addition, it is often the case in real environment that recommender systems are judged in certain domains or contexts. That means different domains often propose different criteria in performance for a recommender system. For example, recommender systems in e-commercial industries may have stricter requirement in coverage, while friends recommendation in social-media platforms may be asked to perform well in accuracy. This fact requires evaluators to compare and contrast recommender systems in a scientific way, not just considering accuracy metrics, but also highlighting beyond accuracy metrics.

Here this chapter carries out several evaluation experiments to comprehensively measure the performance of three collaborative filtering recommendation algorithms, user-based, item-based, and biased matrix factorisation detailed in Chapter 3. The three algorithms are evaluated on two datasets with different data properties. Furthermore, according to the evaluation results, the relationship between the algorithms, the selected metrics and the datasets are discussed as well.

5.2 Experimental Setup

5.2.1 Datasets

There are two datasets used to perform the proposed evaluation, FilmTrust [48] and MovieLens-100k [3]. FilmTrust is a small dataset crawled from the entire FilmTrust[1] website in June, 2011 and MovieLens data sets were collected by the GroupLens Research Project[2] at the University of Minnesota. As shown in Table 5.1, although they are both from the domain of movie, but there are big differences in terms of the statistical properties. For example, the data density (the percentage of rated user-item pairs in all user-item pairs) of MovieLens-100k dataset is almost 6 times as large as that of the FilmTrust dataset which has less rating records as well. Moreover, the scales of the two datasets are slightly different, with the FilmTrust from 0.5 to 4 and MovieLens-100k from 1 to 5. In calculating evaluation metrics, the different scales means if it is necessary to normalize the metrics.

[1] http://trust.mindswap.org/FilmTrust/
[2] https://grouplens.org/

Dataset	Users	Items	Ratings	Scale	Density	Items
FilmTrust	1,508	2,071	35,497	[0.5, 4.0]	1.14	Movie
MovieLens-100K	943	1,682	100,000	[1, 5]	6.30	Movie

Table 5.1: Properties of the Filmtrust and MovieLens-100k datasets

5.2.2 Methodology

The three algorithms are run in the form of test cases using the web application. There are three test cases created of which each is responsible for an algorithm. The evaluation metric selection, evaluation methodology and parameter setups for the three algorithms are easily configured in a test case. Details are given as follows.

First, the evaluation is designed to focus on the offline experiment in main two aspects. One is the task of making predictions and the other one is the task of generating a top-N recommendations. Based on the tasks, the major prediction accuracy metric selected are RMSE (Equation 2.3). On the other hand, for the top-N task, F1 (Equation 2.6) is the major relevance metric used and the beyond accuracy metrics used include popularity (Equation 2.7), diversity (Equation 2.11), novelty (Equation 2.12), and the variation of item space coverage Shannon entropy (Equation 2.10). The two tasks are both run with 5-fold cross validation which randomly splits the dat set into 5 samples. For each sample, one of the 5 samples as the test set is evaluated based on the predictions or recommendations made using the remaining 4 samples as training set, repeating 5 times to guarantee every subsample has been used as the test set. The final result is gained by averaging over the different samples.

Third, the parameter setups on the algorithms. For the three algorithms in recommendation experiment, the size of recommendation lists are given 10, i.e. $N = 10$. For the two-neighbourhood algorithms, user-based and item-based, they share all similar configurations. For both of them, the similarity approach is given Pearson correlation (Equation 3.2) and the neighbourhood size is set by default 50. In the prediction task, the predicted rating is calculated by Resnick algorithm (Equation 3.3). In the recommendation task, the recommended items are ranked by the predicted ratings. For the variation of matrix factorisation, biased matrix factorisation(BMF), Table (Equation 5.2) gives its parameter configurations (where Reg is short for regularisation).

Learn Rate	Iterator No.	Factor	Bias Reg	User Reg	Item Reg
0.01	100	20	0.01	0.01	0.01

Table 5.2: Configurations of biased matrix factorisation(BMF)

Last, there are main two experimental scenarios set up to rigorously compare and contrast the performance between the three algorithms. The first one is to run the task of making predictions to measure how accurate the items are predicted by the algorithms. The second one is to run the task of forming recommendations to analyse the tradeoff between relevance metrics and beyond relevance metrics. The two scenarios are planned out for so-called comparative analysis in the level of different recommendation algorithms.

5.3 Results

Based on the proposed experimental scenarios, the following gives the results and analysis of them.

5.3.1 Prediction Accuracy

Figure 5.1 manifests the result of prediction accuracy metric, RMSE, performed on MovieLens-100k and FilmTrust datasets. Due to the different scales of ratings in the two datasets, the selected RMSE is normalised by being divided by the range of rating scale.

The larger the metric the algorithms get, the poorer they perform in making accurate predications. As seen from the graph, overall, item-based algorithms gain a good performance on both datasets. In each of the datasets, BMF seems to achieve higher scores in RMSE than the user-based and item-based algorithms, telling that the deviation of the item ratings predicted by BMF from the actual ratings is larger. In addition, it is found that the two neighbourhood-based algorithms interestingly performs almost equally in the task of making predictions.

To conclude, taking the movie domain and the datasets into consideration, if a CF recommender system has to be chosen and employed in the context of making predictions, the experimental result encourages to choose the item-based one.

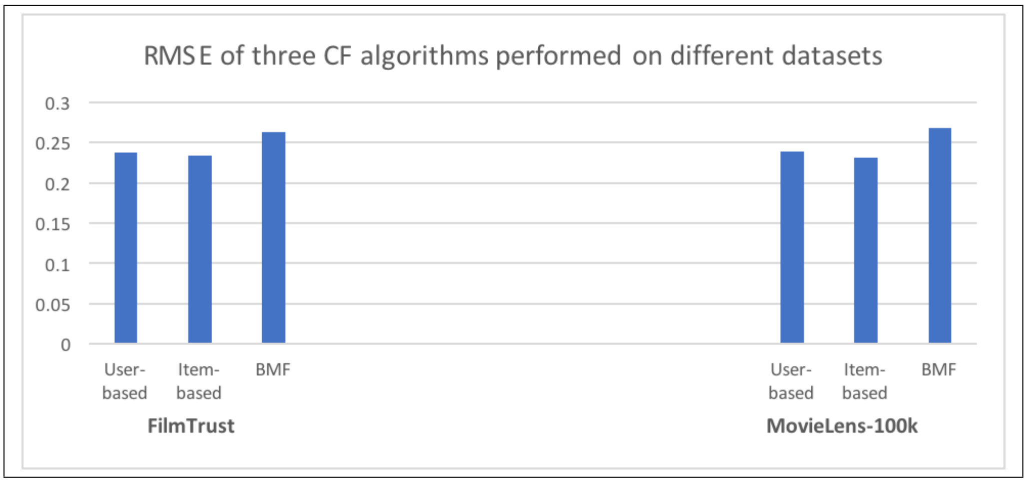


Figure 5.1: Prediction accuracy performed on MovieLens-100k and FilmTrust datasets

5.3.2 Top-N Accuracy

Figure 5.2 shows the result of the relevance metric F-measure in top-10 task, performed on MovieLens-100k and FilmTrust datasets. The F1 produces an outcome between 0 and 1. 0 indicates total irrelevance of the recommended items, while 1 means total relevance of the recommended items.

Unlike the prediction task, it is shown in Figure 5.2 that the different datasets have some impact on the algorithms. Overall, the three algorithms achieve higher F1 score performed on the FilmTrust dataset than the MovieLens dataset. As known from the statistical properties of the two datasets in Table 5.1, the ratings in FilmTrust is more sparse than MovieLens-100k. This infers that one of reasons influencing the three algorithms' performance in recommending relevant items is probably the dataset density.

Taking a closer look at the result, for the FilmTrust dataset, as the F1 metric indicates, BMF performs poorer in suggesting relevant items than user-based and item-based. However, the situation is just the opposite in the MovieLens-100k dataset where BMF achieves the highest F1 score compared to the other two. Also, the result shows that user-based almost gets the same performance as item-based on both datasets (0.2 versus 0.18, and 0.41 versus 0.43).

In summary, performed on either dataset, the performance of the three algorithms in the recommendation task tends to be marginally different. However, performed on different datasets, the performance tends to be slightly different.

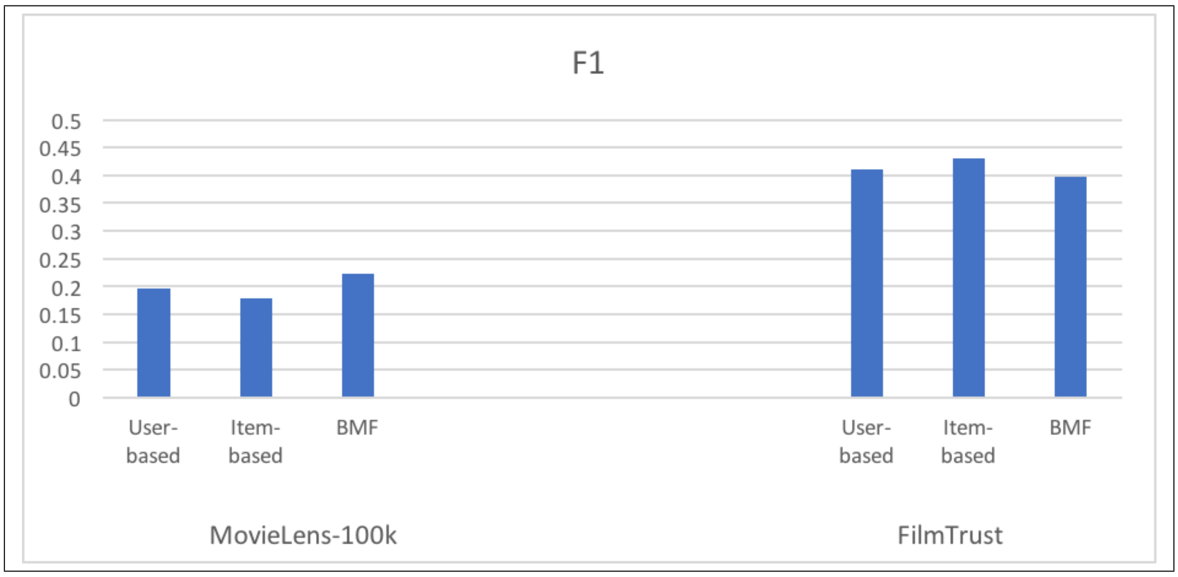


Figure 5.2: Top-10 accuracy performed on MovieLens-100k and FilmTrust datasets

5.3.3 Beyond Accuracy

Table 5.3 and Table 5.4 shows the results of the proposed beyond accuracy metrics for the three algorithms, performed on the FilmTrust and MovieLens-100k respectively. The following gives the analysis for each of them and the relationship between them.

	Popularity	Entropy	Diversity	Novelty
User-based	0.353936504	19.80576326	0.114467484	22.90560346
Item-based	0.348116736	22.5082339	0.114245773	24.34700012
BMF	0.321250044	31.56051898	0.214065647	30.17142389

Table 5.3: Beyond accuracy metrics performed on FilmTrust dataset

	Popularity	Entropy	Diversity	Novelty
User-based	0.330762655	21.15629439	0.09558181	28.15863222
Item-based	0.321655644	38.66905408	0.093588239	30.05523119
BMF	0.297528173	54.07078172	0.166845221	37.94125076

Table 5.4: Beyond accuracy metrics performed on MovieLens-100k dataset

- **Popularity** is an important metric for measuring the performance of recommender systems. If a recommender system is biased towards into popular items, for instance, always suggesting viral videos, it will put constraints on users in gaining new knowledge. In this metric, popularity is calculated by averaging over the popularity of each item in the top-10 list. The lower score the algorithm obtains, the less popular items that are recommended. In order to provide a reference telling what the metric value means, the top-10 most popular items (i.e. top-10 items that have been rated by users the most) are constructed as a recommendation list. It is found that the popularity of the most popular list in the FilmTrust dataset is about 0.5, and in MovieLens-100k is about 0.42. As shown from the Popularity column in the two tables, it is found that user-based and item-based performs almost the same in recommending popular items, both achieving around a popularity score of 35%. By contrast, BMF is a little likely to recommend popular items in each case of the datasets, achieving 32% and 29% popularity in FilmTrust and MovieLens-100k respectively. It is concluded that BMF is the approach that performs the best in recommending less popular items in terms of the popularity criterion.

- **Entropy** is a variation measurement of item-space coverage. It indicates the percentage of items out of the item space that is effectively recommended to a user. Defined by Equation 2.10, this coverage variation is calculated by the entropy of all items in the recommendation lists. Taking the most popular list as an example, if the same popular list is recommended to all users in the system, the entropy score will be 0. As the statistics show in the Entropy column, in both datasets, it is obvious that BMF is capable of recommending a larger portion of items to users than user-based and item-based. Also, it is found that the user-based algorithm performs poorly in the coverage criterion, with the average entropy only being approximately 20 which is far lower than BMF.

- **Diversity** is another important metric telling how diverse the items in the recommendation lists are. If a recommender system is biased towards into similar items, for instance, suggesting many films of the Star Wars series, users will probably think the recommender system is boring. Intuitively, the algorithm that achieves higher coverage is potentially easier to generate diverse recommendations. The experimental results of diversity prove this intuition. This metric produces an outcome between 0 and 1, if all items are the same recommended to users, the diversity will get a value of 0. If all items are totally dissimilar recommended to users (very ideal in real systems), the outcome will be 1. The result in the two tables shows that, among the three CF algorithms, BMF performs the best in recommending diverse items to users (scoring 0.21 and 0.16 in two datasets) and user-based and item-based almost have the same ability to suggest diverse items (nearly 0.11 and 0.09). For different datasets, one interesting finding is that, compared to user-based and item-based, the advantage of BMF in diversity using FilmTrust is more obvious than the counterpart using MovieLens-100k.

- **Novelty** acts as a metric opposite to the popularity metric. It tells the percentages of recommended items that users are not aware of. A recommender system with a good performance in novelty is likely to stimulate users' interest and improve user experience with the system. In this metric, in both datasets, the novelty results correlate with the popularity metrics. In comparison to the popularity, BMF gains a low score in popularity and therefore a high score in novelty (30.17 and 37.94 in both datasets). Besides, the popularity scores tells that user-based is biased towards suggesting popular items. Here its novelty score shows that it is poor at recommending novel items (22.9 and 28.2 in both datasets).

Combined the result in this experiment with the features of each of the three algorithms, here are several points to conclude. First, user-based is severely biased towards popular items, which can be explained by the fact that the algorithm operates over user-user similarities. In user-based approach, items that are popular are rated by a large portion of users and often given high ratings.

This means that popular items can form neighborhoods that contain a larger portion of users that assign high ratings to the popular items and therefore are more likely to be recommended. Unlike user-based algorithms, the similarity computation in item-based is based on items. As found in the experimental results, the problem that item-based suffers from is diversity. Applying the same analysis to item-based, given a certain item, items similar to the item in taste are more likely to become its neighbour items and therefore is easier to be recommended. This well explains why item-based is relatively poor at suggesting diverse items.

As seen from the beyond accuracy results in the proposed experiment, overall BMF has the best performance among the three CF algorithms. It is less biased towards popular items at the same time is able to recommend diverse and novel items. BMF as a model-based approach, there defines a predicted matrix to approximate the actual user-item matrix by learning item and user features in the dataset. In another words, there is a error-conquer process involved in matrix-factorisation approach, detailed in Chapter 3. In matrix-factorisation approach, the mechanism that the prediction matrix learns to approximate the user-item matrix guarantees the relevance of recommended items. The feature of exploration for potentially existing user and item features makes it more likely to perform well in coverage, diversity, and novelty of top-N recommendations.

5.4 Conclusion

Inspired the importance of evaluating recommender systems in a scientific way, this chapter conducts several evaluation experiments to comprehensively evaluate the performance of three CF algorithms. The experimental scenarios are designed based on two tasks, making predictions and generating a top-10 recommendations. In each of the scenarios, two different datasets from the same domain (Movie) but with different data density (1.14% in FilmTrust versus 6.30% in MovieLens-100k) are given. Also, different evaluation metrics are selected to measure the performance in each scenario.

First, the prediction scenario is run and the RMSE metric reflects that user-based and item-based performs almost equally in making predictions, both better than BMF. Next, in order to further study the performance of the algorithms in making recommendations, two categories of metrics are chosen to measure the performance, relevance F1, and beyond accuracy metrics like popularity, diversity and novelty, etc. It is important to measure the recommendation relevance in evaluation experiments because recommender systems will be worthless if they are unable to recommend relevant items to users. However, only emphasizing relevance is insufficient to comprehensively measure the performance of recommender systems. Finally, the algorithms' performance in the beyond accuracy metrics are analysed. Some interesting findings are captured in the section. For example, user-based is biased towards into popular items; item-based is biased towards into diverse items; BMF performs well in the beyond accuracy metrics.

Chapter 6: **Conclusion and Future Work**

6.1 Conclusion

In this project, a general study of the performance of recommender systems is conducted. There are many different recommendation algorithms proposed to meet the requirement of discovering preferred items in a large information space. The algorithms include content-based, collaborative-filtering, hybrid etc. Among these algorithms, collaborative-filtering (CF) algorithms are considered well-performed and the most commonly-applied in modern world. Hence, this project concentrates on the CF algorithms. There are main three types of CF algorithms, user-based and item-based and matrix-factorisation, each of which has different performance in different domains. In the domain of recommender systems, a major challenge is how to evaluate recommender systems comprehensively so as to find the algorithms that best suit a certain domain. Inspired by the challenge, the project presents an outcome of a web application for offline evaluating the three CF algorithms.

This report goes from literature review to a comparative performance analysis of three algorithms. First, the work that has been done in literature and related to the project is reviewed. The emphasis is placed on the research of beyond accuracy metrics. One purpose of the background research is to implement the web application more comprehensively. For example, when considering metrics used for evaluating recommender systems, many beyond accuracy metrics are learnt from the literature and eventually implemented in the application. The other purpose of literature review is to come up with new ideas based upon the work that has done in literature. For example, an extension of coverage measurement named Entropy is introduced to the system.

Next, the reports presents the implementations of the three CF algorithms explicitly, user-based, item-based and matrix-factorisation. The three algorithms are the main focus of the project. For the user-based and item-based algorithms, the implementations normally go the steps from data presentation, similarity computation and neighborhood formation to make predictions or recommendations. For the matrix-factorisation algorithms, the implementation process is described, first initialising the prediction model for capturing latent features in the datasets, and second approximating the model to the existing user-item matrix through iterations using a loss function under updates rules. There are many variations of matrix-factorisation algorithms. One variation named biased matrix factorisation is implemented in web application.

With the technical details of relevant algorithms implementations in the system presented, then the report next delivers the development process of the web application. Main activities in this part include system overview and architecture, front-end and back-end implementation and testing. The overview gives a general introduction to how the compare and contrast between different algorithms are carried out. The architecture presents major components of the system. In terms of the project's outcome, there are some important factors considered in the stage of system architecture, such as the session mechanism, role separation, etc. The front-end implementation section gives good features of the web application on the browser side. The back-end implementation section introduces some technologies that provide support for the server-side implementation. Combining the front-end and back-end implementation, the web application is achieved to support readily configurable for evaluation metrics and extensible for recommendation algorithms. The final section of the chapter goes to testing methodology where some major testing scenarios conducted to test the application are briefly described.

Finally, a chapter is given to present a comparative performance analysis of the three CF algorithm using the web application. The chapter aims to compare and contrast the performance of the three CF algorithms in both accuracy metrics and beyond metrics. Also, a scientific evaluation process is stated in the chapter. In terms of the experimental setups, the algorithms are arranged to perform on two different datasets in two experimental scenarios. In one aspect, the prediction metrics are used to measure their performance in making predictions. In another aspect, the recommendation metrics are used to measure their performance in generating top-N recommendations. In analysis of the experimental results, some interesting findings include that, user-based algorithms are biased towards popular items, item-based perform relatively poor in recommending diverse items, and matrix-factorisation achieve good performance in beyond accuracy metrics.

6.2 Future Work

Although this web application has been well-established to provide a platform for designers to evaluate recommender systems comprehensively using different evaluation metrics, there is still some work to do in the future.

- Expand to support more algorithms. The application now only supports three collaborative filtering algorithms, user-based, item-based and biased matrix-factorisation. As the application is readily extensible for recommendation algorithms, it is expected to include more other algorithms, such as hybrid algorithms, content-based algorithms [35], demographic-based algorithms [38], etc.
- Include more evaluation criteria. In literature, there are so many evaluation metrics proposed to measure the performance of recommender systems. Also, there are also so many different variation of a given metric, for example, many approaches for measuring coverage. Although the current system has got some important metrics implemented, different domains often have different requirements for evaluating recommender systems in terms of performance criteria. Hence, it is necessary to include more evaluation metrics in the system. For example, extra metrics include serendipity [7, 8], learning rate [7], confidence [7], etc.

References

[1] Anderson, C., 2006. The long tail: Why the future of business is selling less of more. Hachette Books.

[2] Herlocker, J.L., Konstan, J.A. and Riedl, J., 2000, December. Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (pp. 241-250). ACM.

[3] Herlocker, J.L., Konstan, J.A., Borchers, A. and Riedl, J., 1999, August. An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 230-237). ACM.

[4] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web (pp. 285-295). ACM.

[5] Ott, P., 2008. Incremental matrix factorization for collaborative filtering. Hochsch. Anhalt, Presse-und ffentlichkeitsarbeit.

[6] Wang, J., Arjen P., De V., and Marcel JT R., 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM.

[7] Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T., 2004. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), pp.5-53.

[8] Ge, M., Delgado-Battenfeld, C. and Jannach, D., 2010, September. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In Proceedings of the fourth ACM conference on Recommender systems (pp. 257-260). ACM.

[9] McNee, S.M., Riedl, J. and Konstan, J.A., 2006, April. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In CHI'06 extended abstracts on Human factors in computing systems (pp. 1097-1101). ACM.

[10] Breese, J.S., Heckerman, D. and Kadie, C., 1998, July. Empirical analysis of predictive algorithms for collaborative filtering. In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (pp. 43-52). Morgan Kaufmann Publishers Inc.

[11] Ricci, F., Rokach, L. and Shapira, B., 2011. Introduction to recommender systems handbook. In Recommender systems handbook (pp. 1-35). Springer US.

[12] Lops, P., De Gemmis, M. and Semeraro, G., 2011. Content-based recommender systems: State of the art and trends. In Recommender systems handbook (pp. 73-105). Springer US.

[13] e Cunha, M.P., Clegg, S.R. and Mendona, S., 2010. On serendipity and organizing. European Management Journal, 28(5), pp.319-330.

[14] Ziegler, C.N., McNee, S.M., Konstan, J.A. and Lausen, G., 2005, May. Improving recommendation lists through topic diversification. In Proceedings of the 14th international conference on World Wide Web (pp. 22-32). ACM.

[15] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P. and Riedl, J., 1994, October. GroupLens: an open architecture for collaborative filtering of netnews. In Proceedings of the 1994 ACM conference on Computer supported cooperative work (pp. 175-186). ACM.

[16] Torres, R., McNee, S.M., Abel, M., Konstan, J.A. and Riedl, J., 2004, June. Enhancing digital libraries with TechLens. In Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on (pp. 228-236). IEEE.

[17] Shardanand, U. and Maes, P., 1995, May. Social information filtering: algorithms for automating word of mouth. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 210-217). ACM Press/Addison-Wesley Publishing Co.

[18] Desrosiers, C. and Karypis, G., 2011. A comprehensive survey of neighbourhood-based recommendation methods. Recommender systems handbook, pp.107-144.

[19] Paterek, A., 2007, August. Improving regularised singular value decomposition for collaborative filtering. In Proceedings of KDD cup and workshop (Vol. 2007, pp. 5-8).

[20] Hanley, J.A. and McNeil, B.J., 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 143(1), pp.29-36.

[21] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2000, October. Analysis of recommendation algorithms for e-commerce. In Proceedings of the 2nd ACM conference on Electronic commerce (pp. 158-167). ACM.

[22] Jarvelin, K. and Keklinen, J., 2002. Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems (TOIS), 20(4), pp.422-446.

[23] Riedl, J., 2009, October. Research challenges in recommender systems. In Tutorial sessions Recommender Systems Conference ACM RecSys.

[24] Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A. and Riedl, J., 2002, January. Getting to know you: learning new user preferences in recommender systems. In Proceedings of the 7th international conference on Intelligent user interfaces (pp. 127-134). ACM.

[25] Aloysius, G. and Binu, D., 2013. An approach to products placement in supermarkets using PrefixSpan algorithm. Journal of King Saud University-Computer and Information Sciences, 25(1), pp.77-87.

[26] Kohavi, R. and Longbotham, R., 2015. Online controlled experiments and A/B tests. Encyclopedia of machine learning and data mining, pp.1-11.

[27] Felfernig, A. and Burke, R., 2008, August. Constraint-based recommender systems: technologies and research issues. In Proceedings of the 10th international conference on Electronic commerce (p. 3). ACM.

[28] Burke, R., 2002. Hybrid recommender systems: Survey and experiments. User modeling and user-adapted interaction, 12(4), pp.331-370.

[29] Koren, Y., Bell, R. and Volinsky, C., 2009. Matrix factorization techniques for recommender systems. Computer, 42(8).

[30] Carenini, G., Smith, J. and Poole, D., 2003, January. Towards more conversational and collaborative recommender systems. In Proceedings of the 8th international conference on Intelligent user interfaces (pp. 12-18). ACM.

[31] Adomavicius, G. and Tuzhilin, A. 2005. Towards the next generation of recommender systems: a survey of the state-of- the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6), pp. 734-749.

[32] Guo, G., Zhang, J., Sun, Z. and Yorke-Smith, N., 2015, June. LibRec: A Java Library for Recommender Systems. In UMAP Workshops (Vol. 4).

[33] Burke, R. and Ramezani, M., 2011. Matching recommendation technologies and domains. In Recommender systems handbook (pp. 367-386). Springer, Boston, MA.

[34] Kim Falk, 2015. Practical recommender systems(pp. 1-20). MEAP.

[35] Adomavicius, G. and Tuzhilin, A., 2015. Context-aware recommender systems. In Recommender systems handbook (pp. 191-226). Springer, Boston, MA.

[36] Lakshmi, S.S. and Lakshmi, T.A., 2014. Recommendation Systems: Issues and challenges. IJCSIT) International Journal of Computer Science and Information Technologies, 5(4), pp.5771-5772.

[37] Bridge, D., Gker, M.H., McGinty, L. and Smyth, B., 2005. Case-based recommender systems. The Knowledge Engineering Review, 20(3), pp.315-320.

[38] Vozalis, M. and Margaritis, K.G., 2004, August. Collaborative filtering enhanced by demographic correlation. In AIAI symposium on professional practice in AI, of the 18th world computer congress.

[39] Koren, Y. and Bell, R., 2015. Advances in collaborative filtering. In Recommender systems handbook (pp. 77-118). Springer, Boston, MA.

[40] Zhuang, Y., Chin, W.S., Juan, Y.C. and Lin, C.J., 2013, October. A fast parallel SGD for matrix factorization in shared memory systems. In Proceedings of the 7th ACM conference on Recommender systems (pp. 249-256). ACM.

[41] Rashid, A.M., Albert, I., Cosley, D., Lam, S.K., McNee, S.M., Konstan, J.A. and Riedl, J., 2002, January. Getting to know you: learning new user preferences in recommender systems. In Proceedings of the 7th international conference on Intelligent user interfaces (pp. 127-134). ACM.

[42] Javari, A. and Jalili, M., 2015. A probabilistic model to resolve diversity-accuracy challenge of recommendation systems. Knowledge and Information Systems, 44(3), pp.609-627.

[43] Makhoul, J., Kubala, F., Schwartz, R. and Weischedel, R., 1999, February. Performance measures for information extraction. In Proceedings of DARPA broadcast news workshop (pp. 249-252).

[44] Castells, P., Wang, J., Lara, R. and Zhang, D., 2011, October. Workshop on novelty and diversity in recommender systems-DiveRS 2011. In Proceedings of the fifth ACM conference on Recommender systems (pp. 393-394). ACM.

[45] Hurley, N. and Zhang, M., 2011. Novelty and diversity in top-n recommendation–analysis and evaluation. ACM Transactions on Internet Technology (TOIT), 10(4), p.14.

[46] Vargas, S. and Castells, P., 2011, October. Rank and relevance in novelty and diversity metrics for recommender systems. In Proceedings of the fifth ACM conference on Recommender systems (pp. 109-116). ACM.

[47] Krug, S., 2000. Don't make me think!: a common sense approach to Web usability. Pearson Education India.

[48] Guo, G., Zhang, J. and Yorke-Smith, N., 2013, August. A Novel Bayesian Similarity Measure for Recommender Systems. In IJCAI (pp. 2619-2625).

[49] Ould, M.A. and Unwin, C. eds., 1986. Testing in software development. Cambridge University Press.

[50] Schein, A.I., Popescul, A., Ungar, L.H. and Pennock, D.M., 2002, August. Methods and metrics for cold-start recommendations. In Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval (pp. 253-260). ACM.

[51] Melville, P., Mooney, R.J. and Nagarajan, R., 2002. Content-boosted collaborative filtering for improved recommendations. Aaai/iaai, 23, pp.187-192.

[52] Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Gupta, S., He, Y., Lambert, M., Livingston, B. and Sampath, D., 2010, September. The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems (pp. 293-296). ACM.

[53] Mangalindan, J., 2012. Amazon's recommendation secret. CNN Money http://tech. fortune. cnn. com/2012/07/30/amazon-5.

[54] Gedikli, F., Ge, M. and Jannach, D., 2011, August. Understanding recommendations by reading the clouds. In International Conference on Electronic Commerce and Web Technologies (pp. 196-208). Springer, Berlin, Heidelberg.

[55] Altman, N.S., 1992. An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician, 46(3), pp.175-185.

[56] Myers, G.J., Sandler, C. and Badgett, T., 2011. The art of software testing. John Wiley and Sons.