Interim report-2.docx

by Gaurav Garg

Submission date: 11-Jan-2022 12:18PM (UTC+0000)
Submission ID: 168414048
File name: Interim_report.docx (254.05K)
Word count: 15098
Character count: 82659

Music Recommendation System Using Deep Learning and Collaborative Filtering Techniques

GAURAV GARG

Interim Report

JANUARY 2022

DEDICATION

My dissertation is dedicated to my family and many friends. Sushil and Rakhi Garg, my beloved parents, whose words of support and drive for tenacity echo in my ears, deserve special thanks. Ayushi, my sister, has never left my side and is one of a kind. This dissertation is also dedicated to my numerous friends who have helped me during the process. I will always be grateful for what they have done for me. I dedicate this work and express my gratitude to my JPMC colleagues and mentor.

ACKNOWLEDGEMENTS

I would like to acknowledge my thesis supervisor Akshita Bhandari for her guidance throughout my thesis work. Having the opportunity to be taught by a subject master has been both a privilege and a delight. Thank you very much for being the kind of supervisor that every student requires. For a young academic, this is the ideal role model.

I would also like to thank my family and friends for encouragement and support over the completion and fulfillment of my research project. I would dedicate my thesis to my parents and sister.

ABSTRACT

With the growing number of facts that humans browse each day, a way to quick acquire information items that meet human's desires has end up an urgent problem in recent times. The effort in facts retrieval have introduced brilliant convenience to those who have a tendency to retrieve facts by getting into a query or keywords. If some statistics-in depth web sites can proactively recommend products or information items that customers can be inquisitive about, it'll substantially improve the performance and satisfaction of customers in obtaining information items. The studies in the field of recommender structures exactly originated on this difficulty. Over the past years, fantastic development has been made into this area, from non-customized to personalised and to more current deep studying recommender structures. Although recommender structures were extensively carried out, there are still many problems and challenges in designing excessive exceptional recommender systems. To degree the quality of a recommender gadget, a systematic and rigorous assessment system is required. This record reviews some current well-set up recommender systems and look at some existing metrics for evaluating them. Besides, this file offers info of the mission's implementation - an internet software for the offline assessment of 3 most important collaborative filtering advice algorithms, item-primarily based, consumer-based totally, and matrix-factorisation. The utility helps an extensive range of quite simply configurable evaluation metrics for customers to visualize the performance between distinct recommendation algorithms. The venture targets to offer a complete platform for designers to evaluate recommender structures and guide them to layout better recommender structures.

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

Chapter 1: Introduction

1.1 Motivation

Almost every information-intensive website nowadays has a recommender system. When a customer browses the target product on Amazon1, for example, a list of likely favoured products is recommended to them [21]. Furthermore, when users are watching a video clip on Youtube2, the system's recommender algorithm [52] proposes some relevant videos to them based on the users' previous actions. In this sense, recommender systems have fundamentally altered how we get information. Recommender systems, as stated in [1,] not just to make it convenient for people to acquire information, but they also have a lot of promise for economic growth. Over the last decade, as more individuals recognise the relevance and power of recommender systems, the search for high-quality recommender systems has remained a hot issue in the community. Many recommender systems have been created and utilised in a range of sectors as a result of ongoing efforts in the field [31]. A crucial concern that arises as a result of this is how to determine the performance of recommender systems so that the best ones can be identified for use in certain settings or domains. The answer is to undertake rigorous and scientific assessment tests to evaluate recommender systems [7, 3, 8, 52]. With the growing use of recommender systems in various applications, evaluating them has become increasingly critical. It is common for an application designer to have to pick among a number of possible recommendation algorithms. Comparing the performance of different algorithms in assessment tests can help attain this aim. Furthermore, analysing recommender systems can aid researchers in the selection, tuning, and design of whole recommender systems. This is due to the fact that while creating a recommender system, certain critical elements determining the system's quality are frequently overlooked throughout the review process. For example, Herlocker et al. emphasise in [7] that when evaluating assessment metrics, assessors should examine not just accuracy measurements, but also certain extra quality indicators, or to put it another way, metrics that value the fact that users are frequently uninterested. The project intends to study the most scientific way for assessing recommender systems, motivated by the necessity of evaluating recommender systems and the emphasis on thorough metric considerations in the assessment trials. The project's implementation is offered in the form of a web application.

1.2 Project scope

Here are a few items to consider when it comes to the project's scope. • All-inclusive. When designing the assessment platform, it is critical to incorporate more comprehensive evaluation measures, particularly beyond accuracy metrics [8, 7]. The word 'comprehensive' here refers to the platform's ability to accommodate a diverse set of evaluation measures. It also emphasises the need of having measurements accessible to compare and evaluate numerous studies. • Evaluation without the use of a computer. To assess recommender systems, there are two primary ways. A/B testing is used in online trials to assess alternative algorithms by monitoring real-world use and feedback [25]. Offline assessment uses real-world rating datasets to test basic recommendation system components [7]. Offline assessment is the focus of this study, as are collaborative filtering methods. Although a variety of recommendation algorithms are well-known and frequently utilised, collaborative filtering (CF) methods [3, 2, 7, 4, 5, 11] are the most widely employed. Despite the fact that the project focuses on CF

algorithms, the built programme is flexible and may easily enable the inclusion of other RS algorithms.

1.3 Objectives and Implementation

The project's overall goal is to provide a decent platform for designers to assess recommender systems and aid them in designing better recommender systems. The major five objectives were retrieved from the project's specification. Every project's Blow includes a summary.

1. A web application. A web application is being created to give users with graphical user interfaces so that they may perform experiments more readily and conveniently. Prototyping, page design, coding, and eventually optimization are all part of the implementation process. The main principle that has been kept in mind throughout the route is to make the front interfaces as engaging as feasible.

2. Implementation of three algorithms. User-based, item-based, and matrix factorisation are the three collaborative filtering techniques. LibRec3 is an open-source library that aids in the implementation. Extending the library and integrating it with the online environment are the tasks at hand.

3. Evaluation visualisation. This goal might be considered the project's centrepiece. First, the system should incorporate three assessment approaches [32]. They are K-fold cross validation, leave-one-out, and repeated random subsampling. Following that, some basic assessment measures are presented (here, largely accuracy measurements). Finally, the comparison and contrast, which involves visualising metrics from numerous studies, should be completed.

4. Session mechanism. The technology has been designed to work in two modes. The first is that a user can access the programme as a guest, implying that there is no connection between the user and the server. The session mode, on the other hand, allows users to log in to the system using a unique session code. Users are in session services supplied by the server in this mode. This goal is not included in the project's scope. Because assessing a recommender system may be time-consuming, the session mechanism is used to handle experiments and store users' running data, such as experiment information, running times, metric details, and so on.

5. Extension and enhancement. The goal is regarded as the project's progressive improvement. The initial component of the extension is included in the dataset. Users can apply datasets in a specific format using a file uploading interface. Furthermore, the most significant expansion extends beyond accuracy measurements. This is the foundation of the project's implementation. Popularity, variety, uniqueness, and coverage are just a few of the indicators that have been expanded and improved. The aforementioned objectives are presented in broad terms, with additional specifics supplied in Chapters 3 and 4.

1.4 Report Structure

• The background of important research to the project is presented in Chapter 2. The power and dimensions of recommender systems are discussed initially, as well as a brief introduction to recommender systems. Then a widely accepted taxonomy of recommendation algorithms is

briefly discussed, including content-based, hybrid, and so on [28]. The discussion of assessment measures for assessing recommender systems is the topic of this chapter.

• The implementations of three collaborative filtering methods, user-based, item-based, and matrix-factorisation, are detailed in Chapter 3. This chapter goes through some of the technical aspects of them.

• The system architecture is discussed initially in Chapter 4, followed by the fundamental components of the system. Second, the chapter delves into the specifics of design thinking and project execution from two perspectives: front-end and back-end.

Chapter 2: Background Research

2.1 Recommender Systems

These days, information burden is becoming more and more of a concern [3]. There are certain recommender systems (RSs) online that recommend appropriate content no matter what information items an user is interested in learning more about [11]. Recommender systems have showed a lot of promise in terms of assisting users in finding interesting and relevant things in a huge amount of data [9]. Recommender systems have evolved into an essential component of today's information filtering systems. RS is a relatively young discipline that arose in the 1990s, compared to many other domains of information filtering systems. Since then, there has been a tremendous growth in interest in the RS field. The popularity of RSS feeds can be attributed to the current era's information requirements. There are a few reasons why recommender systems are so popular and effective. Anderson argues in his well-known work The Long Tail [1] on the relevance of using recommenders in the current society. The author of the essay begins by posing a question, namely, why do blockbusters or popular items with great demand dominate the market, and how does this market have a detrimental impact on the economy? The Long Tail Theory, he goes on to say, is a new marketplace style that emphasises the economic benefits of a vast number of niches in the tail. In other words, focus on the tail, where enormous profits may be made or even current bestsellers can be surpassed by selling less popular items in large quantities. To ensure a world of abundance and avoid the monopoly of popularity on profitability, he advises businesses to implement online streaming stores that support a wider distribution channel of products first, and then great long tail businesses to use lower prices to attract customers down the tail later. Finally, he emphasises the importance of recommender systems, based on the idea. He claims that recommenders are particularly useful in guiding people to previously undiscovered specialty goods in the tail. "A solid tip may simplify consumers' exploration of the unknown while also ending the tyranny of the hit, thereby displaying the power of the long tail," he stated. Anderson's idea encourages more service providers to rely on recommenders in order to boost their turnover rates. In addition, there are bonus points that demonstrate the effectiveness of recommender systems. For starters, recommender systems result in a profit for certain brick-and-mortar businesses [25]. Second, because RSs may propose various items to different users based on their tastes, excellent recommender systems enhance the efficacy of discovering interesting items. For example, the well-developed recommendation engine used on the video-streaming internet platform YouTube [52] suggests videos to viewers based on user-recorded data from the past. Users can locate potentially desired things more readily with the help of the recommender. Third, RSs are crucial in e-commerce organisations like Amazon, where items are recommended to consumers in a personalised or non-personalised fashion in order to encourage them to buy more [11].

2.2 Recommender System Dimensions

There are various characteristics of a recommender system to examine in the literature. The essential dimensions for characterising recommender systems are presented in Blow. In addition to these dimensions, certain aspects related to assessing recommender systems are addressed.

2.2.1 Domain

The domain identifies the qualities of material that is suggested [33]. A top-10 of news or blogpost suggestions in your inbox, videos suggested on a digital video streaming service, goods on some e-commercial channels, places to stay from a house renting internet site, or indeed honeymoon attractions for a newlywed couple [34], are all examples of recommender systems' domains. When creating a recommender system, it is critical to consider the domain. This is because, first and foremost, it establishes the severity of incorrect suggestions for a recommender system [34]. To put it another way,

the impact of appropriate product suggestions on Amazon may be less severe than the impact of pharmaceutical recommendations to patients. Second, according to [33], various recommendation domains provide distinct chances for knowledge collecting and application, resulting in a mapping between domain features and recommendation technologies. Last but not least, it considers if an RS should make repeated suggestions. In the case of music, for example, the tracks that have been suggested to consumers are no longer fit to recommend. This dimension serves as a crucial guideline for selecting metrics in assessment experiments based on the domain's functionalities. To be more explicit, in the case of medical recommendations, assessors must place a greater emphasis on the accuracy metric due to the significant cost of incorrect recommendations in that sector. The movie recommendation example, on the other hand, requires evaluators to assess if the personalized recommendation system is able of proposing a wide range of new and varied movies, putting a greater focus on metrics other than accuracy.

2.2.2 Context

The context is concerned with the setting in which suggestions are made [7, 11]. Contextual information is a notion linked to context that relates to the background information of suggestion receivers, such as the time, mood, place, and weather they are in, or the people that are around them, and so on. Contextual information is useful in some areas and can help recommender systems perform better, such as time and location information for local restaurant recommendations. Because of its relevance, context-aware recommender systems have been well-executed as a subclass of recommendation algorithms to increase the quality of recommender systems. [35], which presents three distinct algorithmic models for adding contextual information into the recommendation process, is a good example. The plan is to assess the various scenarios by analysing real-world usage and comment on recommendations. Because of the ability to interact with users, online testing may be used to assess a recommender system with contextual data inserted. It is difficult to evaluate the efficacy of context-aware recommender systems using offline studies since contextual information is directly tied to the user environment, varies at various times, and must be acquired in real-time and quickly.

2.2.3 Purpose

Another key aspect to consider when describing recommender systems is the objective. The aims for which a proposed approach is utilised can be characterised as the purpose [33]. The goal might be for academics to learn more about recommendation algorithms, or for evaluators to conduct online testing, or simply for internet users to find relevant content. Because the purposes of consumers and information producers might occasionally differ [34], understanding for the goal is frequently vital in genuine marketing. Users, for example, may be more keen to secure high-quality suggestions, whereas suppliers may be more eager to propose specific services or products in order to gain income. According to the article, taking into account all of the user goals makes it easier to pick which assessment metrics to use to assess the quality of recommender systems.

2.2.4 Personalisation

Recommendations at the level of personalisation can be offered at two levels: personalised and non-personalised. • It is not personalised. Users are recommended popular and relevant goods via non-personalised recommender systems. They're frequently seen on a news site, where readers are presented with a list of the top-n most popular stories, or on an e-commerce site, where consumers are recommended mutually-related items, such as Amazon's product association suggestion. Non-personalised recommender systems can suggest or anticipate items that consumers are more interested in than those that are randomly chosen. When compared to personalised recommender systems, one advantage of no personalised recommender systems is that the data model for producing suggestions is simpler. • It's unique. Although non-personalized recommendations are effective in some situations, personalised recommendations are the most researched in this field. Amazon's Recommended For You list [53] is a well-known non-personalised recommendation example, where recommendations are prepared particularly for the intended customer. To put it another way, customised

suggestions are tailored to the tastes or profiles of the user. Personalized recommendations have been shown to have a significant impact on users' decision-making processes [7], which indicates that customised recommenders can have a significant impact on user behaviour such as which item to check first, what things to buy, and which items to pore over in depth. A learning procedure, on the other hand, is likely to aid in the optimization and advancement of personalised recommenders [42]. All types of data or facts about users obtained by a customised recommender may be utilised to improve the next round of suggestions, it is widely accepted.

2.2.5 Input and Output

The interface on which a recommender system creates suggestions is referred to as the input and output of a recommender system [34]. The evaluations or taste preferences that users provide into the system are referred to as input. Netflix, for example, used to allow customers to choose genre and topic selections [34]. When a user expresses his opinion about a particular item, the system is said to have received an explicit input [2, 14]. The video suggestions on Youtube, for example, are created by analysing user behaviour from past encounters with the system, such as clicking on a certain video. Implicit input [2, 14] is the input via which the system tries to infer human taste. The results of recommender systems are included in the output. Users can be provided with output in a variety of ways. It might be a top10 list of personalised suggestions, a list of non-personalized recommendations, or an item prediction. The output is split into two groups in the literature. Recommenders who can explain their suggestions are known as white-box recommenders, whereas those that can't are known as black-box recommenders [2]. The Amazon recommender system, for example, proposes a product to consumers after explaining that they previously purchased a product that is comparable to this one. Explanatory recommender systems have been shown to persuade users to buy and acquire trust in the system as a whole [54, 2].

2.3 Taxonomy of Recommendation Algorithms

There are several well-known recommendation algorithms. They can be categorized as follows in literature. • The focus is on the content. Content-based algorithms [34] are algorithms that create recommendations based on content information and user characteristics. It works by comparing the qualities of the user's profile with the properties of a content item to propose new interesting items to the user [1]. The most significant step in creating a content-based recommendation engine is calculating item-item similarity or mapping things to profiles. The content representation, including extracting significant item contents and expressing them into data structures, frequently plays a part in the computation. Traditional content-based algorithms describe objects in an unstructured manner using information retrieval techniques. Although, for items which are established by clustering, such as commodities on e-commerce sites, a content-based approach known as case-based employs a very well set of features and statistical properties to represent items in a far more structured manner, allowing for fine-grained judgements about item similarity [37]. • Filtering in a group (CF). "When one neighbour assists another, we build our communities," Jennifer Pahlka observed, and this phrase perfectly encapsulates how a CF system operates. In addition, a CF system tries to produce suggestions by merging the preferences of comparable users [3, 2, 7, 4, 5, 11]. Its collaborative filtering property allows it to perform well in customised recommender systems [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.

Nonetheless, some scholars have highlighted certain inescapable limits. Herlocker et al. [2] bring out why a CF system's stochastic computing processes and data sparsity can lead to it not being recognized for elevated content domains like drug recommendation. This is where the project fits under the CF methods. As a result, Chapter 3 delves deeper into the technical specifics of the three CF algorithms in this category: user-based, item-based, and matrix-factorisation. It is a hybrid. Content-based algorithms, as is well known, provide recommendations based on content models, but collaborative

filtering techniques excel in recommending unique and serendipitous items [3]. Hybrid techniques using two or more recommendation algorithms have been proposed to leverage the benefits of suggested algorithm [16, 28]. [28] is an example of integrating CF and content-based techniques to create a hybrid system. Aside from the algorithms above, there have been a slew of new ones developed in recent years. Context-aware recommender systems, for example, offer recommendations based on contextual information [35]. In complicated products like financial services or electronic consumer goods, constraint-based [27] methods are utilised to make the product selection process more successful. Conversational techniques are intended to provide an interactive process between the system and the user in order to gather user input data [30]. Many alternative recommender algorithms that use the user input data just once to create suggestions are unable to overcome the problem of poor comprehension for user preferences because of this aspect [11]. Demographic techniques give suggestions based on user qualities classed as demographic data, such as gender, race, age, and so on [38]. [38] provides an example of how demographic data might be utilised to improve CF algorithms.

2.4 Evaluating Recommender Systems

An overview of many methodologies and metrics for evaluating collaborative filtering recommender systems is presented in this section.

2.4.1 Evaluation

Methodology If a significant number of decision support techniques are chosen to be evaluated, the productivity of online assessment will be poor. As a result, for many existing recommender algorithms, an initial filtering is required. This is how another offline evaluation mechanism [7] is introduced. Offline assessment uses current userrating datasets to assess basic components of recommendation systems. Offline evaluation trials can be conducted using a variety of strategies [32]. Leave-one-out (LOO), k-fold cross validation (KCV), and leave-ratio-out (LRO) are the three methods (LOV). LRO divides the data into training and test sets at random, with a ratio value indicating how much of the training set is used. KCV divides the data set into k samples at random. One of the k samples is used as the test set for each sample, and predictions or recommendations are generated using the remaining k-1 samples as the training set, with the process repeated k times to ensure that every subsample has been utilised as the test set. LOV chooses one user-item-rating as the test data and the rest as the training set, repeating this process for each observation to ensure that every observation has been utilised as the test set. The final results of KCV and LOV are obtained by averaging over a single round of assessment, while the result of LRO is supplied by the measurement score in the test set.

2.4.2 Accuracy Metrics

When it comes to performance evaluation measures, accuracy measurements are the most popular. There are a number of reasons why accuracy measurements are usually taken into account in the first place. First and foremost, accuracy is a critical performance metric that indicates how near received things are to users, i.e., their relevance to their needs. Second, accuracy measurements serve as a useful guide to the measures presented in the next section. Propose that recommender systems deal with a variety of jobs, the two most popular of which are making a forecast and suggesting a list of products, based on end-user duties. The measures for accuracy may be classified into two groups. The prediction metrics are appropriate for the prediction task, although top-N metrics are preferable for the recommendation task.

2.4.2.1 Prediction Accuracy

Mean Absolute Error (MAE): Normally, the dataset is separated into a train set and a test set for offline evaluation of the prediction job. The train set is used to train the prediction model, whereas the test set items are utilised to evaluate performance because actual ratings for the predicted items are known. The difference between the anticipated and actual ratings, intuitively, is the direct approach to gauge the accuracy of the items predicted by a recommender system. Mean absolute error (MAE) is the term for this [10, 7, 17]. Equation 2.1 is used to compute it. MAE = P N |pa,j − ra,j | N (2.1) Where pa,j is the predicted rating of user a for item j, ra,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 (MSE) and Root Mean Square Error (RMSE): Although simple to use and comprehend, MAE does not discriminate between big and small mistakes, usually a large inaccuracies are not condemned. Two options are presented in order to address the situation. They are the mean squared error (MSE) [10, 2] and the root mean square error (RMSE) [10, 2] derived using Equation 2.2 and Equation 2.3, respectively. In the two approaches, the error is squared to penalise large errors. MSE = P N |pa,j − ra,j | 2 N (2.2) RMSE = rP N |pa,j − ra,j | 2 N (2.3)

2.4.2.2 Top-N Accuracy A list of suggestions is provided for the active user after the recommendation task is completed. The challenge necessitates the use of top-N accuracy measures. Top-N accuracy measures, also known as classification accuracy metrics (CAE), are used to assess the relevance of top-N recommendations, as detailed by [7]. The top-N metrics measure how accurate or inaccurate a recommender system's judgments are [2]. CAEs are tolerant of differences in projected and actual ratings, but they are demanding about how effectively the recommender system classifies items, specifically the relevance, locations, and orderings of items in top-N lists. Precision and recall are two categorization measures often used to assess the relevance of top-N suggestions in the field of information retrieval. Recall & Precision: The percentage of relevant things in the top N suggested items is represented by precision [7, 11, 34]. Recall [7, 11, 34] represents the percentage of relevant items that are recommended. It is described in formula as Equation 2.5. P recision = |REL ∩ REC| |REC| (2.4) Recall = |REL ∩ REC| |REL| (2.5) where REC and REL represent the recommended and relevant items in the test set respectively. F-Measure: Because either one of accuracy and recall may be changed independently of the others. For example, when the number of items recommended grows, recall increases while accuracy decreases. The F-measure [43, 21] is a new metric that combines the two metrics into a single score that indicates the relevance of recommended items. F1 [21], which is determined using Equation 2.6, is one of the F-measure implementations. F1 = 2 × Recall × P recision Recall + P recision (2.6) Other approaches: Many different top-N measures have been presented as replacements due to significant limits in accuracy and recall [7]. Mean accuracy precision (MAP), for example, computes the average precision over various levels of recall [2]. Another method, known as receiver operating characteristic (ROC) [20], offers a theoretically based solution to the dilemma of just considering binary significance in accuracy and recall. Furthermore, the normalised discounted cumulative gain (nDCG) [22] emphasises the locations of recommended goods, penalising relevant items in lower rankings.

2.4.3 Beyond Accuracy

Metrics Beyond accuracy measures shift emphasis to additional features that substantially explain the effectiveness of recommender systems [8]. Prediction accuracy metrics assess how close the things predicted deviate from the actual ratings, and top-N metrics indicate how relevant the items are suggested. Popularity, coverage, diversity, uniqueness, and other characteristics are among them. Beyond accuracy measures are difficult to assess because they frequently incorporate levels connected with user subjective views, such as emotions or prejudices [16]. Despite the difficulties, several ways of measuring them have been presented in the literature. The projects beyond accuracy metrics are discussed first, followed by some other out-of-accuracy factors.

2.4.3.1 Popularity

Popularity must be evaluated in terms of understanding the effectiveness of recommender systems. The importance of popularity as a measure describes how well-liked the suggested products are. This means that it is especially critical for e-commerce companies. If a recommender system can only create popular things, as indicated in [1, this indicates that only around 20% of the catalogue area gets examined, leaving the other 80% of niche products unexplored. In real-world testing, a recommender system with a lower popularity score outperforms a system with a higher popularity score when it comes to uncovering potentially controversial things. Averaging over the proportion of each item's popularity is a simple approach to gauge the popularity of products recommended to a user.

Popularity(j) = Nj N (2.7)

2.4.3.2 Coverage

A range of items for which a recommender may make a forecast or offer suggestions is referred to as coverage [2]. Users will notice a significant difference if the recommender has a high converge rate. It has been cited in a number of articles as one of the most essential criteria for evaluating a recommender system [42, 7, 8]. The coverage is frequently characterised in the literature from three angles. For starters, the proportion of target users for whom at least one recommendation may be provided can be calculated. This form of coverage indicates whether or not all users in the system are likely to get suggestions. Second, "what is the percentage of objects for which the system is capable of providing a forecast or recommendation?" may be measured. This is referred to as prediction or suggestion coverage [7, 8]. Out of the item space, this coverage reflects how vast the suggestion space is. The percentage of items in the dataset that appear at least once in the top-N suggestions generated across all target users [8] is a straightforward technique to calculate this sort of coverage. The following is the measuring formula. RecCoverage = IIpI III (2.8), where I is the set of available items and Ip is the set of things that have been suggested to users at least once. Another way to estimate coverage is to ask "how many products are ever recommended to users out of the accessible items?" This is what is known as catalogue or item-space coverage. This sort of coverage is critical for recommender system performance since a recommender system that can fill a considerable part of item space in its suggestion list is considered high-quality [8]. A easy technique to determine this sort of coverage is to average the percentage of things in a specific target user's candidate suggestion list out of all available items across all target users. Equation 2.9 is used to compute it. P uU ItemCoverage = P uU ItemCoverage = P uU ItemCoverage = P Cu III IUI (2.9), where ICuI denotes the number of candidate items for user u, and I and U denote the set of available items and users, respectively. In the procedure stated above, there is a bottleneck. It ignores the number of times each item appears in the candidate recommendations and does not take individual items into consideration. An optimised strategy termed Shannon entropy is offered to solve the problem. The Shannon entropy is determined using Equation 2.10, as stated in [22]. EntropyCoverage = X L i=1 pi log(pi) (2.10), where pi represents the proportion of item I in the recommendation lists and L represents the number of candidate items.

2.4.3.3 Diversity

Diversity, in contrast to similarity, is a metric that is typically used to the top-N list of recommendations to determine how varied the things recommended are [14, 44, 7]. Recommender systems that propose a variety of items can prevent users from exploring repetitive and pointless objects, which is inefficient for the user job of identifying all useful items [7]. Pairwise dissimilarity [14], which computes the similarities between all pairs of items in the top-N recommendation list, is the most widely employed technique to quantify diversity in relevant research. The method is shown in Equation 2.11.

Diversity(r1, ..., rN) = P i=1...N P j=1...N∧i6=j (1 − sim(ri , rj))) N(N − 1) × 2 (2.11)

where the diversity is determined by the top-N suggestions Sim(ri, rj) is the similarity between item I and j, which may be estimated using similarity functions such as the content-based similarity function, the cosine similarity function, or Pearson correlation, as discussed in Section 3.1.

2.4.3.4 Novelty

Novelty is another essential factor to consider when looking beyond accuracy numbers. It assesses a system's capacity to recommend objects to users who are unaware of them [44, 3, 45, 8]. It's a significant metric since absence of innovative recommendations in recommender systems causes consumers to be unaware of the presence of some goods in the catalogue [22]. Novelty is frequently linked to popularity, or, to put it another way, to the polar opposite of popularity. Self-information-based novelty [46], for example, is a measure of novelty that posits that popular goods give less novelty. The formula is as follows: S(N) = P UUT estSet (PN i=1 log(IUI relsi)) IUT estSetI) (2.12) where UT estSet is the test

set's set of users, IUI is the number of test users, and relsi is the number of users who have rated item i. When it comes to novelty, it's worth noting that suggestions won't make sense if they're only new but irrelevant to the user [7]. As a result, originality should be examined in conjunction with accuracy measures, as well as other non-accuracy criteria.

2.4.3.5 Other Beyond Accuracy

Many different metrics have been presented in the literature to describe aspects of recommender systems other than accuracy. Serendipity is a metric for the ability to propose goods that are unexpected and appealing to consumers [13, 8, 7]. The distinction between serendipitous and new items is that serendipitous items are less likely to be discovered by users, whereas novel goods are generally those that users can locate but are unaware of. The learning rate indicates how quickly an algorithm can create appropriate suggestions [7], and privacy is used to quantify the extent to which recommenders employ privacy-protected data and demonstrate to a user that it is secure.

2.5 Conclusion

This chapter began with an introduction to multiple dimensions of recommender systems, as well as some critical factors for assessing recommender systems within these dimensions. The taxonomy of recommendation algorithms was then briefly described. Finally, the underlying study focused on the assessment of recommender systems, with two types of evaluation metrics presented: accuracy metrics and beyond accuracy measurements. The methods presented in this chapter will be useful for the project's implementation and experimentation analysis, which will be covered in Chapters 4 and 5. The three collaborative filtering algorithms used in the research will be the topic of the following chapter.

Chapter 3:

RESEARCH METHODOLOGY

Collaborative Filtering Algorithms

There are two primary subclasses of collaborative filtering (CF) algorithms according to the taxonomy of recommendation algorithms: neighbourhood-based and model-based. User-based [17] CF and item-based [ ] CF are two neighborhood-based techniques. Both methods attempt to locate products or people that are comparable to the currently active person or item. Model-based techniques entail extracting key data from datasets and using that data as a model for making suggestions. Matrix-factorisation [5, 39] is a well-known method in this family. The specific implementations of the three CF algorithms are listed below.

3.1 User-based

User-based [17] methods discover people with similar likes to the active user and give suggestions or predictions. The following stages must be addressed in order to construct a user-based CF algorithm.

3.1.1 Data Representation

The preference data of users for things must be represented before the similarities between users can be calculated. The preference data is provided here as explicit ratings on a scale of 0 to 5 for simplicity and clear presentation. Table 3.1 depicts an example of data visualisation in which users' evaluations for products are shown as a matrix. The matrix is made up of five rows and four columns. Each column represents an item designated i1... i4 and each row represents a user labelled u1... u5. The numbers N and I denote the number of users and objects, respectively. User i's rating for item j is represented by the number in row I and column j. For instance, the u5 rating for i2 is 2. Furthermore, the table's blank cells indicate that the goods have not been rated by the users.

	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

The next stage is to compute the similarity between the active user and all other users in the system once the user preference data has been collected. In user-based algorithms, this is a critical stage in constructing the relationship model between users. Many methods for computing similarity have been proposed in the literature, including mean square difference and extensions such as significance weighting and Inverse User Frequency. Two typical approaches for calculating user similarity are presented below. The vector/cosine similarity is calculated using the equation below. In the M-dimensional item space, users are represented as vectors, and the similarity between users a and I is determined as the cosine of the angle between user a and user i's vectors. 1 indicates total agreement on co-rated items, 0 implies no similarity between users and -1 indicates total disagreement. wa,i = P j∈Ia∩Ii (ra,j − ra)(ri,j − ri) qP j∈Ia∩Ii (ra,j − ra) 2 P j∈Ia∩Ii (ri,j − ri) 2 (3.2) where ra refers to the mean of all ratings given by user a. Following the similarity computation, each user compiles a list of similarities with other users.

3.1.3 Neighbourhood Formation

After the similarities have been computed, the next step is to create a neighbourhood. To put it another way, the current user's neighbourhood should be chosen from the top most comparable users. There are several methods for forming a neighbourhood. K nearestneighbour (KNN) [55] is a frequently used method for forming a neighbourhood of size k by picking the k users with the highest similarity to the current user. The neighbourhood size k is usually determined by trial and error. It must be chosen carefully since a big k will diminish accuracy (the impact of the most similar neighbours will be diluted), but a small k will result in poor accuracy (users will not have enough close neighbours).

3.1.4 Make a prediction or generate a top-N list

Following the development of a neighbourhood, the active user's expected rating of an item is computed by taking a normalised weighted average of the item's ratings from each of the user's neighbours. Resnisk's formula [15, 3] is a widely used method for estimating expected ratings. The expected rating of user a for item j is computed using Equation 3.3. This approach

$$pa,j = ra + Pk\ i=1\ wa,i(ri,j - ri)\ Pk\ i=1\ |wa,i| \quad (3.3)$$

The similarity score between users a and I is wa,i, and the size of the neighbourhood is k. In order to generate a top-N list, one method is explained as follows: To begin, create a candidate group of things for suggestion by adding all of the goods that have been rated by the active user's neighbours together. Second, the collection is filtered to exclude things that have already been rated by the active user. Finally, the remaining things are ordered using the projected ratings derived using Equation 3.3, and the top-N ranked items are returned to the target user as recommendations.

3.2 Item-based

In user-based CF algorithms, there is a bottleneck. User-based CF algorithms have significant scalability issues [23]. In real-world systems with possibly millions of users and objects, calculations are costly, as seen in the issue. This is how item-based algorithms, which detect linkages between objects rather than users, are introduced to alleviate the bottleneck. Item-based CF algorithms [4] are founded on the premise that users are interested in things that are similar to those they have already encountered. The item-based implementation follows the same four phases as the user-based implementation, as described in Section 3.1. The data representation stage is same. Pairwise similarities are generated based on objects rather than users in similarity computation. Items are represented as vectors in the N-dimensional user space in the cosine similarity technique, for example, and the similarity between item a and item j is determined as the cosine of the angle between item a and item j. As a result of this phase, a data structure will be created to hold the similarities between objects instead of users. An item neighbourhood is created by selecting a subset of other things that have the most resemblance to the current item. In making a prediction, taking the Resnisk's algorithm as an example, the prediction equation 3.3 is changed to

$$pa,j = vj + Pk\ i=1\ si,j\ (ra,i - vi)\ Pk\ i=1\ |si,j\ | \quad (3.4)$$

where si,j is the similarity score between items I and j, vj is the mean rating of item j across all users, ra,i is user a's rating of item I and k is the neighbourhood size. The procedure for creating a top-N list of recommendations is the same as that detailed in Section 3.1.4 for user-based recommendations.

3.3 Matrix Factorisation

Both user-based and item-based CF algorithms have various drawbacks. First, in order to make predictions or recommendations, neighborhood-based algorithms significantly rely on the relationships between people and goods. This shows that the relationship model is difficult to construct due to a lack of data (the similarities between neighbours). The data sparsity problem is what it's termed. Second,

they have an issue with early raters. Predictions for things that are freshly added to a system are difficult to make since the items are rarely rated in the system (not enough ratings to calculate predictions). It also shows that making predictions or suggestions for people who have just enrolled is difficult because they haven't shown enough interest in the system (not neighbours). Model-based techniques are established as a result of these restrictions. There are several well-established model-based techniques, the most popular of which being matrix factorisation [5, 39]. It operates by assuming that there are some hidden characteristics that explain user preferences. Its purpose is to reveal these hidden qualities, which may then be utilised to better precisely anticipate goods for a user. It begins by setting up the number of features K and the two matrices P (IUI K matrix) and Q (III K matrix), which indicate the strength of the link between a user (P) or an object (Q) and each of the K characteristics. The prediction matrix, which displays the predicted ratings, is then calculated by multiplying P and QT.

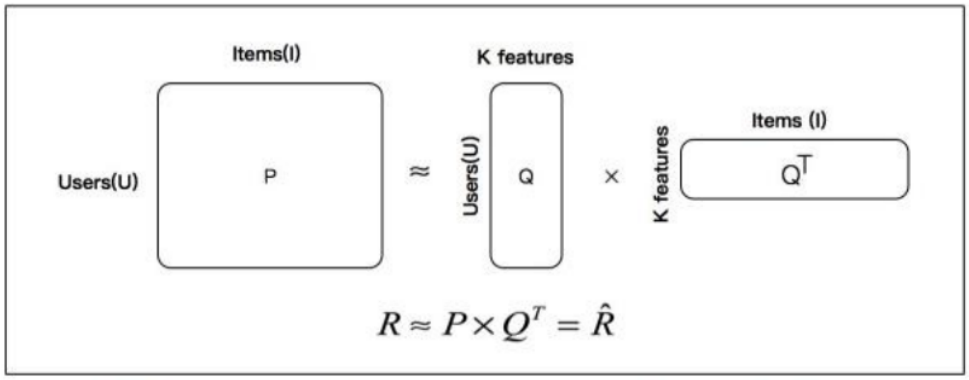


Figure 3.1: Illustration of matrix factorisation whose goal is to find the predicted rating matrix $\hat{R}$ by learning user and item feature values

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,

$$r\hat{}i,j = p\ T\ i\ qj = PK\ k=1\ pikqkj$$

Iterations are allocated to update P and Q in order to approximate the real user-item matrix, and the difference, termed the loss function, is supplied by the error between the predicted and actual ratings. The mistake can be calculated in a variety of ways. Square error loss is one example [18]. The square error over all ratings in the training set is calculated using the equation below.

$$T.\ E = X\ ri,j\in T\ e\ 2\ i,j = X\ ri,j\in T\ (ri,j - X\ K\ k=1\ pikqkj\ )\ 2 \quad (3.5)$$

After getting the loss function, a choice must be taken about how to change P or Q to reduce the error. Many ways have been developed to attain this aim, including Simon Funk1's stochastic gradient descent (SGD) [40]. It operates by adjusting the values of pik and qkj in response to the gradient's direction. It loops through each rating 1 in the training set T until the error is minimised, updating the values of pik and qkj each time using update rules for each factor. The updating rules for both are given in Equation 3.6.

$$\text{pik and qkj .}$$
$$p\ 0\ ik = pik + \alpha\ \delta\ \delta pik\ e\ 2\ i,j = pik + 2\alpha ei,jqkj\ q$$
$$kj = qkj + \alpha\ \delta\ \delta qkj\ e\ 2\ i,j = qkj + 2\alpha ei,jpik \quad (3.6)$$

where α is the learning rate normally given by 0.001.

The discrepancy between the prediction matrix and the actual user-item matrix is minimised using the loss function and update rules (SGD). The preceding discusses a fundamental step in the use of matrix

factorization techniques. There are several ways to implement matrix factorisation algorithms in the literature. For instance, the linear matrix factorisation (LMF) strategy presented by [19, 5] works by adding two bias constants and regularisation parameters to the prediction model to avoid overfitting and hidden user biases. In this method, user i's projected rating for item j is altered to

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

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

It's worth noting that the two constants, pik and qkj, must be learned simultaneously. Normally, they are delivered by 0.01. The same procedure as described previously in the section is used to train the predicted matrix R as near as feasible to the real prediction model by learning the user-item matrix R using the loss function and update rule. In this project, the LMF method is implemented as a version of matrix factorisations, although it is called biassed matrix factorisation (BMF). The next chapter goes into further depth regarding its performance.

3.4 Experimental Setup

3.4.1 Datasets

The suggested evaluation is based on two datasets: FilmTrust [48] and MovieLens100k [3]. The MovieLens data sets were gathered by the GroupLens Research Project2 at the University of Minnesota, while FilmTrust is a tiny dataset scraped from the whole FilmTrust1 website in June, 2011. Although they are both from the movie domain, as demonstrated in Table 5.1, there are significant variances in terms of statistical features. For example, the data density of the MovieLens-100k dataset (the fraction of rated user-item pairs in all user-item pairs) is over 6 times that of the FilmTrust dataset, which includes less rating records. Furthermore, the sizes of the two datasets varied somewhat, with FilmTrust ranging from 0.5 to 4 and MovieLens100k ranging from 1 to 5. The different scales indicate whether or not it is essential to normalise the measurements when computing evaluation metrics.

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 3.4.1: Properties of the Filmtrust and MovieLens-100k datasets

3.4.2 Methodology

The web application is used to execute the three algorithms as test cases. Three test cases have been established, each of which is accountable for a different algorithm. In a test case, you may quickly configure the assessment metric, evaluation technique, and parameter configurations for the three algorithms. The following are the specifics. First, the assessment is geared on focusing on the offline trial in two key areas. The first is to make predictions, and the second is to generate a top-N list of suggestions. The key prediction accuracy statistic used is RMSE, which is based on the tasks (Equation 2.3). For the top-N task, on the other hand, F1 (Equation 2.6) is the most important relevance measure, with popularity (Equation 2.7), diversity (Equation 2.11), novelty (Equation 2.12), and Shannon entropy variation of item space coverage also being employed (Equation 2.10). Both tasks use 5-fold cross validation, which divides the data set into 5 samples at random. One of the five samples is used as the test set for each sample, and it is assessed based on the predictions or recommendations produced using the remaining four samples as the training set, with the process repeated five times to ensure that every subsample has been used as the test set. The final result is obtained by averaging the results of the several samples. The parameter settings on the algorithms are the third step. The size of recommendation lists for the three algorithms in the recommendation experiment is provided as 10, i.e.

N = 10. All of the setups for the two-neighbourhood algorithms, user-based and item-based, are the same. The similarity method is provided Pearson correlation (Equation 3.2) for each of them, and the neighbourhood size is set to 50 by default. The Resnick method is used to calculate the predicted rating in the prediction job (Equation 3.3). The anticipated ratings are used to rank the suggested items in the recommendation job. Table (Equation 5.2) shows the parameter configurations for biassed matrix factorisation (BMF), a version of matrix factorisation (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)

Finally, two key experimental situations have been built up to evaluate and contrast the performance of the three algorithms. The first is to perform the prediction task to see how accurate the algorithms are in predicting the items. The second is to conduct the process of making suggestions in order to examine the tradeoff between relevance measures and metrics that are not relevant. The two situations are set up for a so-called comparative comparison of various recommendation systems.

3.5 Prediction Accuracy

The outcome of the prediction accuracy metric, RMSE, for the MovieLens100k and FilmTrust datasets is shown in Figure 5.1. 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 bigger the measure, the worse the algorithms perform in terms of producing accurate predictions. Overall, item-based algorithms perform well on both datasets, as seen in the graph. BMF appears to attain higher RMSE scores in each dataset than the user-based and item-based algorithms, indicating that the variation of the item ratings predicted by BMF from the actual ratings is greater. Furthermore, it is discovered that the two neighborhood-based algorithms perform nearly equally well in the task of producing predictions.

To summarise, if a CF recommender system is to be chosen and used in the context of producing predictions, the experimental result suggests choosing the item-based one, taking the movie domain and datasets into account.

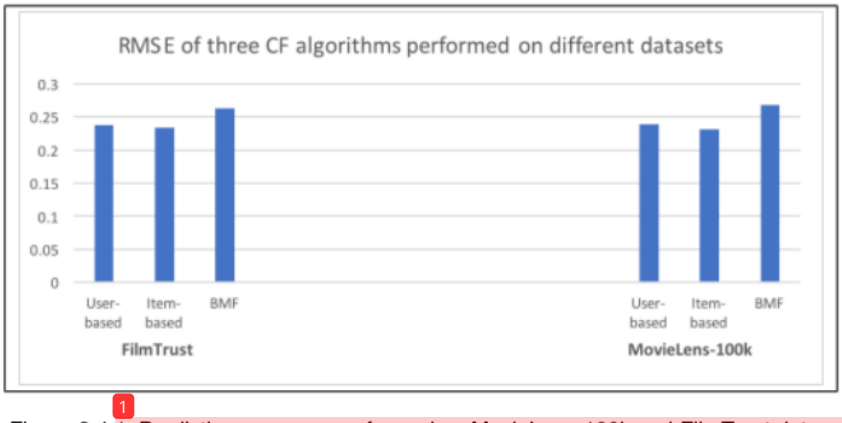


Figure 3.4.1: Prediction accuracy performed on MovieLens-100k and FilmTrust datasets

3.5.1 Top-N Accuracy

On the MovieLens-100k and FilmTrust datasets, Figure 5.2 displays the outcome of the relevance metric F-measure in the top-10 job. The F1 yields a result between 0 and 1. Entire irrelevance of the recommended things is indicated by a value of 0, whereas total relevancy is shown by a value of 1. Unlike the prediction job, the diverse datasets have an influence on the algorithms, as seen in Figure 5.2. On the FilmTrust dataset, the three algorithms get a better F1 score than the MovieLens dataset. The ratings in FilmTrust are sparser than MovieLens-100k, as shown by the statistical features of the two datasets in Table 5.1.

This implies that one of the factors impacting the effectiveness of the three algorithms in proposing relevant items is most likely dataset density. Taking a deeper look at the results, the F1 statistic shows that BMF performs worse than user-based and item-based in proposing suitable items for the FilmTrust dataset. In the MovieLens-100k dataset, however, the situation is reversed, with BMF receiving the greatest F1 score of the three. In addition, the results reveal that on both datasets, user-based obtains about the same performance as item-based (0.2 versus 0.18, and 0.41 versus 0.43).

In conclusion, when the three algorithms are tested on the same dataset, their performance in the recommendation job is minimally different. However, when tested on other datasets, the results tend to vary slightly.

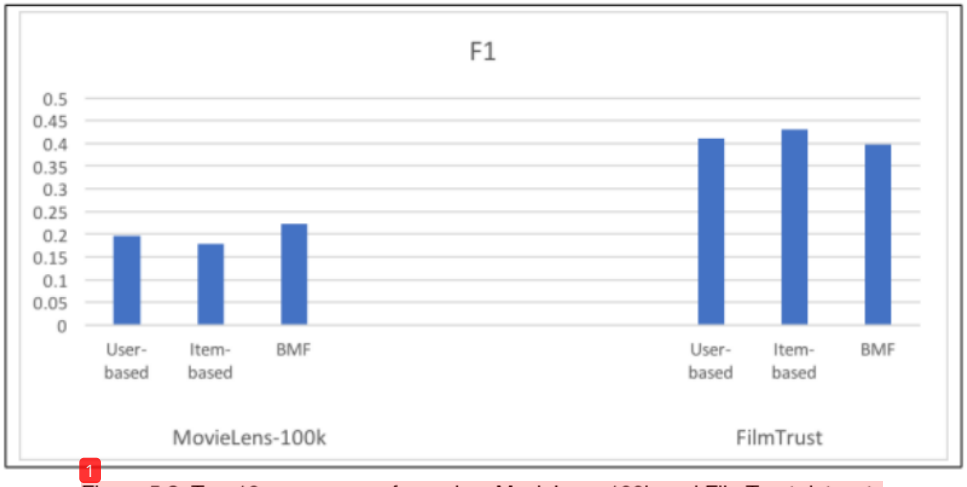


Figure 5.2: Top-10 accuracy performed on MovieLens-100k and FilmTrust datasets

3.6 Conclusion

This chapter undertakes numerous assessment tests to comprehensively analyse the performance of three CF algorithms, inspired by the significance of assessing recommender systems in a scientific manner. The experimental scenarios are built around two tasks: making predictions and coming up with a top-10 list of suggestions. Two alternative datasets from the same domain (Movie) are provided in each scenario, but with differing data density (1.14 percent in FilmTrust versus 6.30 percent in MovieLens-100k). In addition, multiple assessment measures are used to assess each scenario's performance. The prediction scenario is first run, and the RMSE measure shows both user-based and item-based prediction perform almost equally well, both better than BMF. Next, two types of metrics are chosen to assess the success of the algorithms in producing recommendations, relevance F1 and

beyond accuracy measurements like popularity, diversity, and novelty, among others. In evaluation studies, it's critical to quantify suggestion relevance since recommender systems are useless if they can't recommend relevant things to consumers. However, stressing relevance alone is inadequate to assess the effectiveness of recommender systems in a complete manner. Finally, the algorithms' performance in terms of measures other than accuracy is examined. The section contains some intriguing results. For example, user-based is biassed toward popular products, whereas item-based is biassed toward diversified goods, and BMF excels in the measures that go beyond accuracy.

3.7 Future Work

Despite the fact that this online application has been well-established as a platform for designers to completely analyse recommender systems using various evaluation criteria, there is still some work to be done in the future.

- Increase the number of algorithms supported. User-based, item-based, and biassed matrix-factorisation are the only collaborative filtering techniques currently supported by the programme. Because the application's recommendation algorithms are easily extendable, it's likely to contain more algorithms in the future, such as hybrid algorithms, content-based algorithms [35], demographic-based algorithms [38], and so on.
- Increase the number of assessment criteria. There are several assessment measures offered in the literature to quantify the performance of recommender systems. There are also several variations of a particular measure such as numerous techniques of measuring coverage. Despite the fact that the present system includes several key measures, various domains frequently have distinct performance criteria for assessing recommender systems. As a result, more assessment measures must be included in the system. Serendipity [7, 8], learning rate [7], confidence [7], and so on are examples of additional measures.

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.

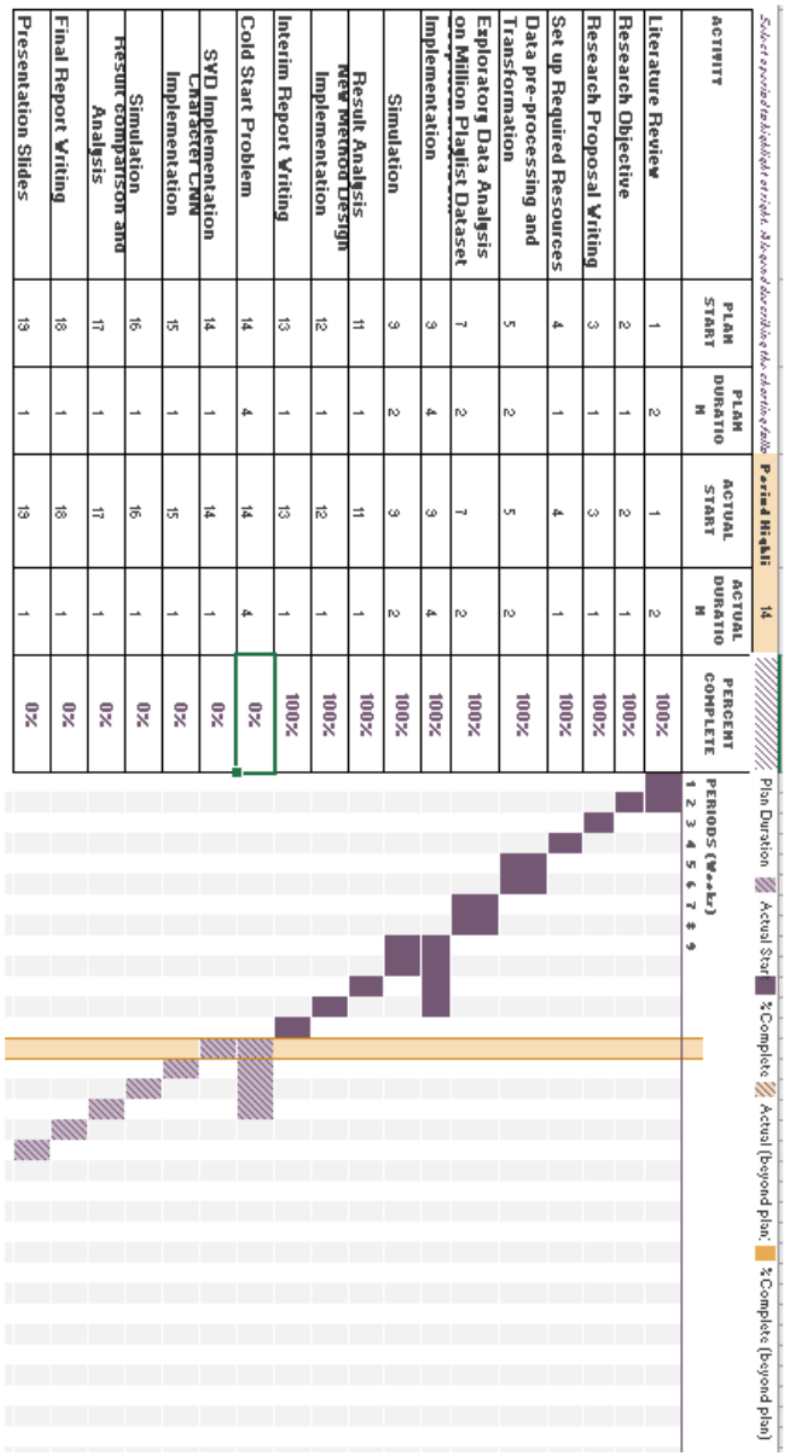
[19] Paterek, A., 2007, August. Improving regularised singular value decomposition for collaborative filtering. In Proceedings of KDD cup and workshop (Vol. 2007, pp. 5-8).

[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.

APPENDIX A: RESEARCH PLAN



ACTIVITY	PLAN START	PLAN DURATION	ACTUAL START	ACTUAL DURATION	PERCENT COMPLETE
Literature Review	1	2	1	2	100%
Research Objective	2	1	2	1	100%
Research Proposal Writing	3	1	3	1	100%
Set up Required Resources	4	1	4	1	100%
Data pre-processing and Transformation	5	2	5	2	100%
Exploratory Data Analysis on Million Playlist Dataset	7	2	7	2	100%
Implementation	9	4	9	4	100%
Simulation	9	2	9	2	100%
Result Analysis	11	1	11	1	100%
New Method Design Implementation	12	1	12	1	100%
Interim Report Writing	13	1	13	1	100%
Cold Start Problem	14	4	14	4	0%
SVD Implementation Character CNN Implementation	14	1	14	1	0%
	15	1	15	1	0%
Simulation Result comparison and Analysis	16	1	16	1	0%
	17	1	17	1	0%
Final Report Writing	18	1	18	1	0%
Presentation Slides	13	1	13	1	0%

APPENDIX B: RESEARCH PROPOSAL

Abstract

Managing and looking for songs has become increasingly important as digital music formats have grown in popularity. Though successful music information retrieval (MIR) algorithms have been developed in the previous ten years, music recommender systems are still in their infancy. As a result, this study examines a broad framework as well as cutting-edge techniques to music recommendation. With the growth of the internet in recent decades, it has become the primary source for retrieving multimedia material such as video, literature, and music, among other things. People regard music to be a significant part of their lives, and they listen to it on a regular basis. Participants listened to music more than any of the other activities, according to previous study. Additionally, the music recommender will assist users in filtering and discovering songs based on their preferences. A smart music recommendation system should be able to detect preferences automatically and produce playlists based on them. Meanwhile, the advent of recommender systems gives the music industry a fantastic opportunity to gather users who are interested in music. More importantly, it presents us with new challenges in terms of better understanding and modelling customers' musical preferences. We explore the current state of the art in solving these problems and its limits. We go over potential future directions and visions for the field's future development. The first stage of our model is tailored for quick retrieval, while the second stage re-ranks recovered candidates to maximise accuracy at the top of the suggested list.

Table of Contents

1. Background

Recommender systems have become a necessary component of many e-commerce and Internet-based organisations such as Google, YouTube, Facebook, Netflix, LinkedIn, Amazon, and others have used recommender systems as part of their business strategies since the Web's inception and spectacular expansion over the last decade. firms in various domains like songs, movies, articles, shopping, etc. These systems are based on user sentiments about a product or service with the goal of promoting them to other users with similar interest according to their past behaviours and tastes. YouTube, for example, offers recommendations for a user based on their viewing patterns and searches for like users. The task of automatic playlist continuation (APC) is for music recommendation systems to find tracks that fit inside a specified playlist. A big part of the APC mission is to appropriately deduce the playlist's intended purpose. This is difficult not only because of the wide range of intended purposes (assuming they exist at all), but also because of the wide range of underlying attributes or characteristics that may be required to infer those purposes. A recommender system is an essential and appealing domain in information filtering systems that seeks to forecast active users' choice or rating of a particular item. A recommender system is a sophisticated engine that uses a variety of ways to suggest the most relevant products for a group of users based on their prior actions and preferences.

Therefore, relevant and appropriate recommendations are extremely important for these businesses to attract more users into using their platform for longer period of time. Even before the digital revolution, music services relied on a variety of entry points into the music catalogue: genres, decades, hit choices, new releases/trending, what's curators/influencers, playlists by context (moods/activities), and provided means for sharing content and playlists. In contrast to other creative content, a song is a three-minute experience, and the question of what to listen to next keeps coming up. As a result, the album's historic format, which ensures a minimum permissible duration, as well as its aesthetic objective, is preserved. Listeners now have devices, apps, and algorithms thanks to the digital revolution, which allow them to better capture those listening situation opportunities and adapt to each context: rich user interfaces on PCs, simplified user interfaces on mobile phones and in cars, voice control in the car, and smart speakers. Listeners can experience rich navigation even when interaction with screens is limited, to the point of inviting designers to create a zero interface where no interaction is possible.

Higher granularity (the ability to provide numerous types of playlists, similar artists and songs), higher frequency of selection updates, a much deeper dive into the collection, and personalisation are all advantages of digital. Various platforms like Rapcaviar, Discover Weekly provides listeners with customised playlist updated every week. Spotify on the other hand, creates multiple playlists based on users most played artist, popular songs in the area and lots of other playlist categorised by genres.

The collaborative filtering algorithm has been determined to function well based on users' listening behaviour and past ratings. When combined with the use of a content-based model, the user can get a list of songs that are comparable based on low-level acoustic variables like rhythm and pitch, as well as high-level features like genre and instrument. Last.fm2, Allmusic3, Pandora4, and Shazam5 are 4 music discovery services that have successfully used these two methodologies. Meanwhile, these websites give a one-of-a-kind platform for retrieving rich and usable data for user studies. There is a lot of possibility for future expansion thanks to new discoveries in psychology, signal processing, machine learning, and musicology. As a result, this study examines a broad music recommender framework, including user profiling, item modelling, and item-user profile matching, as well as a number of state-of-the-art methodologies. We conclude and suggest a new model based on user motivation at the end of this study.

2. Problem Statement

In the last decade, recommender systems have been in the focus for many studies to develop new methods and improve their accuracy. Two types of recommender systems are: collaborative filtering and content-based filtering methods. Most recommender systems are based on collaborative filtering as it is feasible and easy to implement. There are two categories of collaborative filtering: memory-based (Li et al., 2016) and model-based strategies (Paradarami et al., 2017). Model-based methods use ratings to train a model and predict ratings on unrated items by users, whereas memory-based methods use similarities between items and users to predict. Researchers (Zarzour et al., 2018) introduced a model-based music recommendation system which clusters users using k-means clustering and reduce dimensionality with SVD. One of the earliest model-based collaborative filtering methods was proposed 2 decades ago (Chen and George, 1999), which was based on the idea that users (or judges) who tend to give similar ratings can be modeled as having identical rating probabilities.

Content-based methods take into account both the profile of user’s interest and description

of item. Bayesian classifiers, for example, are used to assess the likelihood of a user loving an item based on its content (Pazzani and Billsus, 2007) and another research (Chen et al., 2008) introduced contest-based recommender systems that employs profitability into CF-based systems. A text similarity method was introduced (Robertson and Walker, 2000) that sets a threshold for text similarity in item description and employs adaptive filtering. These systems suffer from many challenges such as misinterpret songs that rarely appear, completing playlist with very few tracks, neglecting sequence of tracks in a playlist. To address these challenges, there are hybrid methods proposed (Nicholas, 1999) and (Ungar and Foster, 1998) which combines collaborative features and content together. Recently, researchers (Xiong et al., 2018) proposed a deep learning-based hybrid method by combining collaborative filtering and textual content. Furthermore, explainable recommendation has also been a field of research interest (Zarzour et al., 2020) where trustworthiness score which depends on users' reactions after watching recommended videos is combined with collaborative filtering.

These collaborative filtering and content-based methods are also applied in music recommender systems. Some authors (Chen et al., 2012) and (Mcfee and Lanckriet, 2011) have developed ways for modelling the transitions between songs in playlists using Markov chains. Although, recently researchers have found that sequence of songs hardly matters to listeners while the song-to-song transitions (Kamehkhosh et al., 2018) and ensemble of songs in playlist (Chen et al., 2018) seems to matter. An online competition (ACM RecSys Challenge 2018) was held for the task of Automatic playlist continuation (APC), which involves recommending songs that are likely to be added to an existing playlist. For example, the recommended songs for a list of new year songs should be other new year songs. As part of the challenge and to promote this type of research at scale, Spotify publicly released the Million Playlist Dataset (MPD) in 2018. The data is now publicly available at (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021) . Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every track. Playlists are filtered to remove tracks with offensive content.

The methods used by participating team in the challenge were mostly based on collaborative filtering combined with dimensionality reduction or CNN models. For instance, (Yang et al., 2018) proposed a 2-phase collaborative filtering model where in first

phase autoencoders is used to encode and decode playlist features and CNN is used for training just playlist titles. Another team (Teinemaa et al., n.d.) used a three-stage model, where in first stage 4 collaborative filtering models are built on 4 context track-track, word-track, album-track and artist-track. First stage is then followed by taking weighted sum of all 4 models and then completing playlist for a given number of tracks based on same album and popular tracks. The winning team (Volkovs et al., 2018) also applied a two-stage model with additional node for cold start problem. Cold start problem occurs when there are only 2 playlist features available: name & length of the playlist. The first stage employs a WRMF collaborative filtering method which is one of most known methods for implicit/binary collaborative filtering (Hu et al., 2008), since WRMF ignores order of songs in the playlist it is clubbed with a temporal CNN and Neighbour-based CF models (song-song, playlist-playlist). The second stage contains an xgboost classifier used to re-rank prediction from 1st stage and additional playlist features like average song/artist/album, song features like artist/album, pairwise playlist song features like similarity between target song and songs in the playlist and furthermore, creative features like danceability, energy, mode etc. taken from spotify audio api.

With the recent improvement in deep learning techniques, many types of powerful news-based (Okura et al., 2017), apps-based (Cheng et al., 2016) and video-based (Covington and Adams, 2016) recommendation systems are based on deep learning. To address the issue of data sparsity and cold-start recently many deep learning techniques are proposed. Another strong technique, Collaborative Deep Learning (CDL), is given, which uses Bayesian formulation to do deep representation learning for the ratings matrix. CDL allows for two-way communication between content information and ratings matrix feedback.

However, there is no study that demonstrates how to embed people and products together. with taking order of songs into account. Also, there are no studies that combine recent deep learning models with cold start problem to build a versatile collaborative filtering model. For example, by clubbing a neighbour-based collaborative filtering model with deep learning and employing a CNN trained on playlist titles to solve for cold-start problem can produce a more effective result. There are also no studies that take number of times a song appears in playlist into consideration.

3. Research Questions

The aim of this research is to develop a recommender system that performs Automatic Playlist Continuation (APC). Since playlist continuation is required to be done in production environment, the recommender system should be versatile enough to work on various limitations such as a smaller number of tracks in the playlist, only playlist title etc. and hence will be tested on such a test set.

Based on the literature review, below research questions have been formulated:

- What are the most and least important playlist and track features during APC?
- What are the best performing models to recommend songs for playlist?
- What solves cold-start problem for playlist with very less features and tracks?
- What can be laid as a foundation for future work in music recommendation systems?

4. Aim and Objectives

The research aims to understand factors influential in music recommendation systems with the task of Automatic Playlist Continuation. The research targets development of an efficient music recommender system to be used by online music streaming websites and eventually help listeners find more tracks they may like.

Below are the research objectives:

- To perform literature review of researches in recommendation systems.
- To perform exploratory data analysis on track and playlist features, find correlation between them and perform feature generation.
- To employ deep learning, clustering and collaborative filtering models for Automatic playlist completion.
- To compare results and identify most efficient models in creating a music recommender system.
- To consolidate results from finalized models.

5. Significance of Study

The study is directed to develop a music recommender system that can be used in production by music streaming platforms to give useful and relevant recommendations to its users. This study will focus on developing a solution that combines recent deep learning models with cold start problem to build a versatile collaborative filtering model which as not been explored yet. The research would also be able to answer which features contribute most and least to music recommendation, this can also greatly help streaming platforms to better understand profile of the user. This study introduces a new collaborative filtering recommendation algorithm based on dimensionality reduction and clustering techniques with the goal of increasing the program's recommendation performance. Music recommendation is relatively a new area of research and is gaining researchers interest over

the years since the business of online music streaming platforms entirely depend on their ability to keep user attracted to the platform for as long as possible and at the heart of it sits the recommendation system.

6. Scope of the study

Recommendation is all about broadening a listener's musical horizons beyond what they already know and enjoy. It gives listeners more navigation speed once they've exhausted all of their song/artist search capabilities. Scope of this study is to make use of the existing deep learning, collaborative filtering and clustering algorithms and to develop new such algorithms that best suits the recommender system. The evaluation of model would be carried out on the separate test set which has playlists of varying lengths, the evaluation metrics used can tell us about recall and top k accuracy of the model. The suggested models will be trained and tested to get consolidated table and arrive at a conclusion on which models performs the best and if there is a scope of further development in the models.

7. Research Methodology

7.1 Introduction

Automated Playlist Continuation (APC) is getting more popular as people listen to music online. While there has been substantial progress in recent years, the majority of the methodologies presented are tested on exclusive datasets, making open collaboration and convention impossible. By undertaking standardised assessments of playlist continuance trends, the 2018 ACM RecSys Challenge intends to close this gap. At the centre of this ambition is the Million Playlist Dataset (MPD), which was launched with the support of Spotify. MPD is the largest publicly accessible dataset of its sort, with over 2.2 million songs and 1 million playlists. The study's goal is to create a playlist continuation version that can suggest suitable following songs for each playlist. For the evaluation, a different set of 10K test playlists is used, with a few songs withheld. The duration of the test playlists ranges from zero (cold start) to a hundred tracks. This mimics a real-world scenario in which the APC model must perform correctly at all degrees of playlist completion. This study's methodology is mostly based on a two-level structure. The first stage seeks to build a big selection of tracks with high recall value for each playlist, and the second stage rates only the tracks generated from first set and select top k. Below are the steps involved for this research:

Step1: Data Cleaning & Understanding

Data understanding and cleaning is important part of a machine learning research. In most research works, data is collected from publicly available sources, which in this case is the MPD published by Spotify and available publicly as an open contest at AiCrowd. Data cleaning steps will be carried out on MPD like removing data sparsity, filling or removing missing values, feature reduction, correlation analysis between features etc.

Step 2: Exploratory Data Analysis

Exploratory Data Analysis (EDA) is carried out to gain hidden insights from the dataset. Univariate, bivariate and multivariate analysis will help in understanding relation between multiple features w.r.t target feature. For visual examination, various charts will be used such as bar chart, heat map, scatter plot, etc. These charts will help to understand trends between different features of the dataset.

Step 3: Pre-processing and Transformation

Pre-processing is required to make data ready for modelling. In the pre-processing steps categorical features will be transformed using one-hot encoding. Features data-type will be restored to the one supported by machine learning model. In addition, outliers will be found and removed from every column in the dataset.

Step 4: Models:

Deep learning, Clustering and Collaborative Filtering are some models to be trained for this research. Every model will be tuned using grid search cross validation. The approach followed will be to first build a baseline model and keep iterating the parameters until we achieve the best results. Different models like xgboost, random forest etc. will also be explored to make sure we use best fit model.

Step 5: Evaluation:

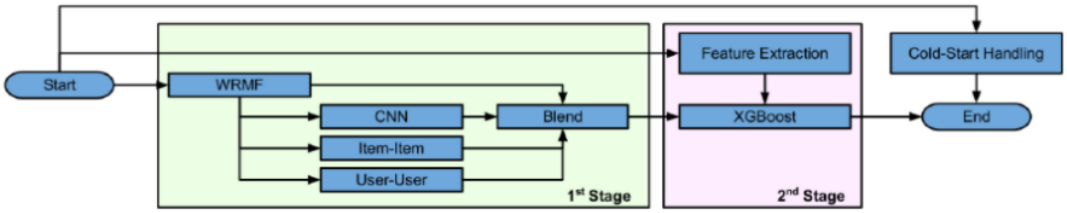
Evaluation is done on test dataset containing 10k playlists of varying lengths. Three different metrics for evaluation are used to test different features of the solution: Recommended songs clicks, Normalized discounted cumulative gain (NDCG) and R-precision. More detail about each of the evaluation metric and what they test is present in "Evaluation Metrics" below.

Step 6: Results & Conclusion

Consolidated results from deep learning, clustering, xgboost and collaborative filtering models will be generated. These results will give insights about which set of models

performs the best and will be compared with actual test set results available to evaluate at AiCrowd (AIcrowd | Spotify Million Playlist Dataset Challenge | Dataset_files, 2021).

Figure 1: Process Flow Diagram



7.2 Dataset description

This research will consume the Million Playlist Dataset (MPD) published by Spotify and available publicly as an open contest at AiCrowd. This data is appropriate for the research because it contains playlist of varying lengths and cold-start problem as well. Spotify being the primary choice for online music streaming for many users all over the world, it has ample number of tracks for all popular languages. This collection of 1 million playlists, drawn from Spotify's over 4 billion public playlists, has over 2 million distinct tracks by approximately 300,000 artists, making it the world's largest publicly available dataset of tracks and playlists. Between January 2010 and November 2017, public playlists generated by Spotify users in the United States were included in the dataset. Every playlist has a title, last modified time, description, total duration, list of tracks with details like track IDs, artist name, track uri etc. for every track. Playlists are filtered to remove tracks with offensive content. Below table has description for features of playlist and track.

Table 1: Playlist attributes

Attribute	Description	Type
pid	The MPD ID of this playlist is its playlist id. This is a number that ranges from 0 to 999,999.	Integer
name	This is the title of the playlist.	String
description	This is the playlist's description. Note: Because for playlist created by users, description is Spotify's new feature, the majority having missing description.	String
modified_at	When was the last time the playlist was updated? Times are rounded to nearest hour on date the playlist was last updated, which is midnight GMT.	Timestamp

num_artists	The number of distinct artists represented in the playlist's songs.	Integer
num_albums	The total number of albums that each track in the playlist has.	Integer
num_tracks	The playlist's total amount of tracks.	Integer
num_followers	The number of people who followed this playlist when the MPD was generated.	Integer
num_edits	The total number of times playlist was edited. Tracks updated into playlist within a two-hour time frame are assumed to be part of a single edit.	Integer
duration_ms	In milliseconds, the duration of all the tracks added in playlist.	Numeric
collaborative	Whether the playlist is collaborative or not. A collaborative playlist can have tunes contributed by multiple people.	Boolean

Table 2: Track attributes

Attribute	Description	Type
pid	The MPD ID of playlist the track belongs to. This is a number that ranges from 0 to 999,999.	Integer
track_name	The title of the song.	String
track_uri	The track's URI on Spotify.	String
album_name	The album's name is the title of the track.	String
album_uri	The album's URI on Spotify.	String
artist_name	The major artist on the track's name	String
artist_uri	The primary artist's Spotify URI for the track.	String
duration_ms	The track's duration in milliseconds	Numeric

pos	The track's position on the playlist.	Integer

This challenge's test set consists of ten separate playlist groups, each having 1,000 playlists. Each group represents a distinct task in terms of completing a playlist:

i. Recommend songs for a playlist based solely on its title
ii. Recommend songs in a playlist based on the title and first track.
iii. Given a playlist's title and the starting five songs, predict the playlist's tracks.
iv. Recommend songs for a playlist based on the first five songs (without title)
v. Recommend songs for a playlist based on the title and the first 10 songs.
vi. Recommend songs for a playlist based on the first ten songs (without title)
vii. Recommend the first 25 songs for a playlist based on the title.
viii. Recommend songs for a playlist based on its length.
ix. Recommend the first 100 songs for a playlist based on the title.
x. Given a playlist's title and 100 random music, predict the tracks for it.

The purpose of this division is to imitate the playlist continuation model's production environment. These playlists will put the model's ability to handle all stages of playlist construction to the test. We will also create a validation set similar to the test set by sampling playlist from train set.

7.3. Data pre-processing and Transformation

In this step we pre-process the data and make it ready for modelling. First step to pre-process MPD is to remove tracks and artist that appear very rarely, because analysing rare samples does not provide statistically significant patterns. For reducing this data sparsity, we filter out tracks that appear less than 6 times and artists that appear less than 4 times. After pruning, the number of tracks is 26% of original count which is 598,293 and number of artists is 155,942 (53%). We would also validate our model on full data to test the hypothesis that if these rare samples provide any significant improvement in accuracy. For playlist titles, we set maximum length to 25 and include only alpha-numeric as well as few special characters (/<>+-).

7.4. Models

We will focus on modelling deep neural networks that include an input layer, a collection of hidden layers, and an output layer in this research. Deep learning model will be used with scores/utility from neighbour-based collaborative filtering models. For activation ReLu function may be used because of its simplicity in optimization, in addition exploring Sigmoid and tanh functions as well. For solving cold-start problem, we plan to explore 2 approaches: dimensionality reduction with SVD or CNN for playlist titles only. Adam optimizer can be used for the deep learning model and CNN model with learning rate of

0.005, which will be tuned using cross validation. The size of filters for CNN model can be 3,5,6 and 9. For Collaborative filtering, 4 neighbour-based (track-track, word-track, album-track, artist-track) or WRMF model can be used. These set of models are expected to produce best results as they are state-of-the-art and are proven to perform better than older ones. Algorithms will be redefined and updated to best suit the use case and will be compared with original to see if they bring any significant changes in accuracy.

7.5. Evaluation metrics

The R-precision, defined as:

$$R - precision = \frac{|T \cap R|}{|T|}$$

where T is the set of ground truth (holdout) tracks, and R is set of recommended songs. The notation |.| denotes the number of elements in the set.

The Normalized discounted cumulative gain (NDCG), defined as:

$$NDCG = \frac{DCG}{IDCG}$$

where:

$$DCG = rel_1 + \sum_{i=2}^{|R|} \frac{rel_i}{log_2(i+1)}$$

$$IDCG = 1 + \sum_{i=2}^{|G|} \frac{1}{log_2(i+1)}$$

The Recommended Songs CLICKS metric, that mimics a Spotify feature for track recommendation where ten tracks are presented at a certain time to the user as the suggestion to complete the playlist. These metric captures required number of clicks or refreshes before a relevant track is found, and is defined as:

$$CLICKS = \frac{argmin_i\{R_i : R_i \in G\} - 1}{10}$$

While NDCG and CLICKS were calculated based on the track-level agreement between the holdout tracks and the recommended tracks, R-precision was calculated on the artist level agreement. In other words, it was considered sufficient if the artist of a recommended track matched the artist of a holdout track.

8. Required Resources

Required hardware and software resources for this research are:

Hardware or Environment Requirements:

- Data Storage – ASW S3 or Google Drive

- Version control system – Bitbucket or Github
- A Laptop
- Processor – Intel Core i3 or higher
- RAM – 6 GB or more
- GPU – Nvidia Graphics Card or equivalent
- OS – 64-bit Windows 10/macOS Catalina or equivalent
- Kernel - AWS Sagemaker or Google Colab or equivalent

Software Requirements:

- Python
- Python libraries
    - Pandas, Numpy, Seaborn, Matplotlib, Scipy, Statsmodel, Keras, Scikit-learn and Tensorflow
- Anaconda
- Jupyter
- Git

* Note: all latest available versions

Interim report-2.docx

ORIGINALITY REPORT

23%	22%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	wangcongcong123.github.io Internet Source	20%
2	Submitted to University of Greenwich Student Paper	2%
3	b2b.musicovery.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		