Evrak Tarih ve Sayısı: 24.04.2023-318

T.C.

AKDENİZ ÜNİVERSİTESİ

Analyzing The Effects of Popularity Bias On Beyond-Accuracy Recommendation Quality In User-Neighborhood-Based Collaborative Filtering Algorithms

Osman Alper MISIRLI

FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği

ANABİLİM DALI

YÜKSEK LİSANS

Şubat 2023

ANTALYA

Bu belge, güvenli elektronik İmza ile imzalanmıştır.
Evrak sorgulaması https://turkiye.gov.tr/ebd?eK=5543&eD=BS9HLYA44Z&eS=318 adresinden yapılabilir. (PIN:66572)

Evrak Tarih ve Sayısı: 24.04.2023-318

T.C.

AKDENİZ ÜNİVERSİTESİ

Analyzing The Effects of Popularity Bias On Beyond-Accuracy Recommendation Quality In User-Neighborhood-Based Collaborative Filtering Algorithms

Osman Alper MISIRLI

FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği

ANABİLİM DALI

YÜKSEK LİSANS

Şubat 2023

ANTALYA

Bu belge, güvenli elektronik İmza ile imzalanmıştır.
Evrak sorgulaması https://turkiye.gov.tr/ebd?eK=5543&eD=BS9HLYA44Z&eS=318 adresinden yapılabilir. (PIN:66572)

Evrak Tarih ve Sayısı: 24.04.2023-318

ÖZET

Kullanıcı-Komşuluk-Tabanlı İşbirlikçi Filtreleme Algoritmalarının Popülerlik Ayrımcılığını Analiz Etme

Osman Alper MISIRLI

Yüksek Lisans, Bilgisayar Mühendisliği

Danışman: Doç. Dr. Alper BİLGE

Yardımcı Danışman: Dr. Öğr. Üyesi Emre YALÇIN

Şubat 2023; 63 sayfa

Bu tezde, kullanıcı komşuluğu tabanlı ortak filtreleme algoritmalarında bulunan popülerlik ayrımcılığını incelemeyi amaçlamaktayız. Popülerlik ayrımcılığı, tavsiye sistemlerinin popüler öğeleri daha sık önerirken daha az popüler öğeleri önermemesi sonucu önerilerde çeşitlilik eksikliğini ifade etmektedir. Bu fenomeni inceleme amacı ile, Kosinüs benzerliği, Ortalama karesel fark ve Pearson korelasyon katsayısı gibi üç farklı benzerlik metrikleri kullanarak KNNwithMeans, KNNwithZscore ve KNNBaseline gibi üç farklı algoritmalayı test ediyoruz. Analizimizi Movielens-1M, Yahoo Music ve Douban Books gibi üç farklı veri kümesinde yapıyoruz.

Çalışmamızın sonuçları, farklı benzerlik metrikleri ve algoritmaların kullanılmasının tavsiye edilen popülerlik ayrımcılığının seviyesini ciddi şekilde etkileyebileceğini göstermektedir. Ayrıca bulgularımız, popülerlik ayrımcılığını azaltmak için farklı benzerlik metrikleri ve algoritmaların bir kombinasyonunun en iyi yaklaşım olduğunu sunmaktadır. Bu yaklaşım, daha çeşitli bir tavsiye kümesi sağlar ve bu daha kişiselleştirilmiş ve memnuniyet verici bir kullanıcı deneyimi yaratabilir. Çalışmamız, tavsiye sistemlerinde popülerlik ayrımcılığının etkisini ve çeşitli yöntemlerin etkililiğini değerli bilgiler sunmaktadır.

Sonuç olarak, bu tez tavsiye sistemlerinde popülerlik ayrımcılığının önemini aydınlatmaktadır ve bunu azaltmak için bir çerçeve sunmaktadır. Bu çalışmanın sonuçları, daha çeşitli ve kişiselleştirilmiş tavsiye sistemleri geliştirmek için kullanılabilir ve bu daha iyi bir kullanıcı deneyimi yaratabilir.

ANAHTAR KELİMELER: Entropi, Ortak Filtreleme, Popüler Tavsiye Edilen Öğelerin Ortalama Yüzdesi, Popülerlik Ayrımcılığı, Uzun Kuyruk Kapsamı, Uzun Kuyruk Öğelerinin Ortalama Yüzdesi

JÜRİ: Doç. Dr. Alper BİLGE

Doç. Dr. Alper Kürşat UYSAL

Dr. Öğr. Üyesi Alper ÖZCAN

Bu belge, güvenli elektronik İmza ile imzalanmıştır.
Evrak sorgulaması https://turkiye.gov.tr/ebd?eK=5543&eD=BS9HLYA44Z&eS=318 adresinden yapılabilir. (PIN:66572)

Evrak Tarih ve Sayısı: 24.04.2023-318

ABSTRACT

Analysing Popularity Bias of User-Neighborhood-based Collaborative Filtering Algorithms

Osman Alper MISIRLI

MSc Thesis in Computer Engineering

Supervisor: Assoc. Prof. Alper BİLGE

Cosupervisor: Asst. Prof. Emre YALÇIN

February 2023; 63 pages

This thesis aims to study the popularity bias present in user-neighbourhood-based collaborative filtering algorithms. Popularity bias refers to the tendency of recommendation systems to recommend popular items more frequently than less popular items, resulting in a need for more diversity in recommendations. To examine this phenomenon, we test three algorithms, KNNwithMeans, KNNwithZscore, and KNNBaseline, using three different similarity metrics: Cosine, MSD and Pearson's corelation coefficient. We analyse three distinct datasets: Movielens-1M, Yahoo Music, and Douban Books.

The results of our study indicate that the use of different similarity metrics and algorithms can significantly impact the level of popularity bias present in the recommendations. Furthermore, our findings suggest that the best approach to mitigate popularity bias is to use a combination of different similarity metrics and algorithms. This approach allows for a more diverse set of recommendations, which can lead to a more personalised user experience. Our study provides valuable insights into the impact of popularity bias on recommendation systems and the effectiveness of various methods for addressing it.

This thesis highlights the importance of addressing popularity bias in recommendation systems. The results of this study can be used to guide the development of more diverse recommendation systems, and personalised, in turn, can lead to a better user experience.

KEYWORDS: Average Percentage of Long Tail Items, Average Percentage of Popular Recommended Items, Collaborative Filtering, Entropy, Long Tail Coverage, Popularity bias

COMMITTEE: Assoc. Prof. Alper BİLGE

Assoc. Prof. Alper Kürşat UYSAL

Asst. Prof. Alper ÖZCAN

Bu belge, güvenli elektronik İmza ile imzalanmıştır.
Evrak sorgulaması https://turkiye.gov.tr/ebd?eK=5543&eD=BS9HLYA44Z&eS=318 adresinden yapılabilir. (PIN:66572)

ÖNSÖZ

Tezin hazırlanması esnasında ve yüksek lisans öğrenimim boyunca maddi ve manevi desteğini esirgememiş olan aileme, kız arkadaşıma, sevgili tez danışmanım Alper Bilge'ye ve yardımcı danışmanım Emre Yalçın'a en içten teşekkürlerimi sunuyorum.

AKADEMİK BEYAN

Yüksek Lisans Tezi olarak sunduğum “Kullanıcı-Komşuluk-Tabanlı İşbirlikçi Filtreleme Algoritmalarının Popülerlik Ayrımcılığını Analiz Etme” adlı bu çalışmanın, akademik kurallar ve etik değerlere uygun olarak yazıldığını belirtir, bu tez çalışmasında bana ait olmayan tüm bilgilerin kaynağını gösterdiğimi beyan ederim.

……./……../20....

TABLE OF CONTENTS

ICONS AND ABBREVIATIONS

K-nearest neighbors: KNN

Mean Squared Deviation: MSD

Pearson’s Corelation Coefficient: Pearson

Yahoo!Music: YM

Douban Books: DB

MovieLens-1M: MLM

Average Percentage of Popular Recommended Items : APRI

Average Percentage of Long Tail Items : APLT

Long Tail Coverage : LTC

FIGURES LIST

TABLE LIST

1. INTRODUCTION

Recommender systems have been used to interact with vast, intricate information spaces. They create a tailored view of such scopes, highlighting items that users are expected to find appealing. Literature in recommender systems has grown significantly since its inception in 1995, with an increasing range of issues being addressed, new techniques being employed, and practical applications being developed. Research in this area incorporates a variety of artificial intelligence methodologies such as constraint satisfaction machines, learning, and even data mining. Personalised recommendations are a central feature of many e-commerce websites like Netflix, Google, Microsoft, Apple, and Sony; the abundance of practical experience gained from these applications has led to researchers expanding the scope of recommendation systems to include new and challenging areas (Burke et al. 2011).

This research focuses on the issue of popularity bias in recommendation systems. In general, popularity bias relates to the problem that recommendation systems prefer some popular items and do not give most other things the attention they deserve. In particular, the purpose of this study is to assess the impact of recommendation systems on creating popularity bias by utilising different parameters and different types of algorithms to define a recommender system for a given recommendation system platform, such as algorithm type or the type of similarity metric to create user-based recommendations.

1.1 Recommender Systems

Search engines assist users in locating information by filtering the most pertinent content from a vast collection of documents, articles, or products. Users express their needs for the type of information they seek by inputting a query, which can take the form of a list of words or other descriptions. In contrast, a recommendation system does not rely on an explicit question. Instead, it utilises its knowledge of the user to predict which items are most relevant, reducing the cognitive load of selection and leaving the final choice to the user(Ricci et al. 2011). Recommendation systems are sophisticated tools and algorithms that aid users in finding content that aligns with their preferences, particularly when the options may be overwhelming. These systems are commonly used to provide personalised recommendations, reducing the user's decision-making burden and simplifying the process of selecting items. A recommendation system can be considered a personal shopper that pre-selects the most appropriate items based on its knowledge of the user, allowing the user to choose from a narrowed-down selection simply.

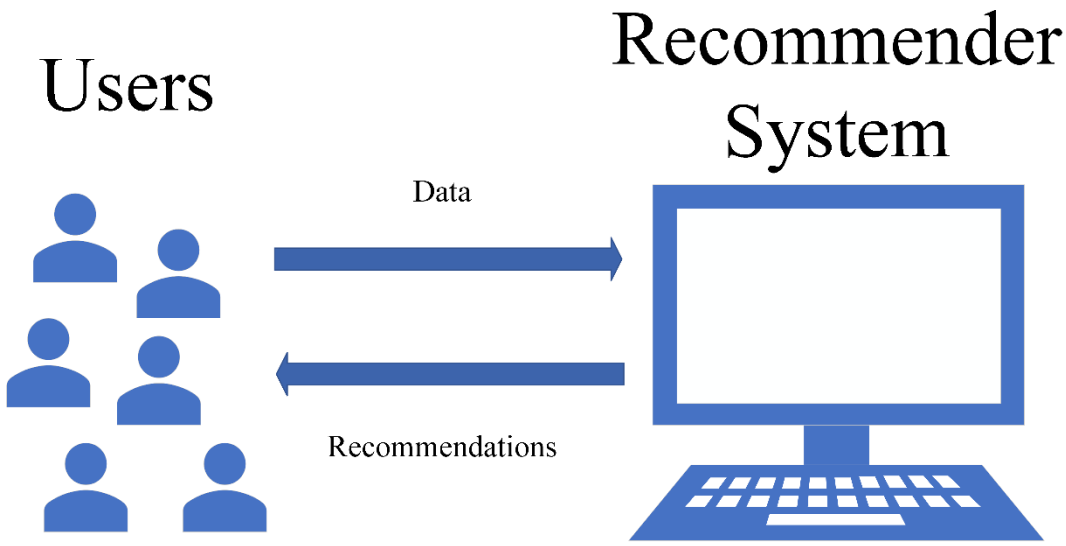


Figure 1.1. Basic recommendation system

In Figure 1.1. the Recommendation System is simplified. A Recommendation system first collects data from the user in the system, such as rating data and similarity data between other users and recommends an item depending on the user's data.

Recommendation systems have matured for many years, but the practice of making recommendations is even older. People commonly rely on the advice of others when making choices in their daily lives. For instance, people may ask for their friends' opinions on things such as films, songs, video games, literature, and employment opportunities. We are more likely to trust these suggestions when they come from someone knowledgeable in the area who shares similar preferences with us or has a track record of giving reliable advice. This is called "word of mouth" communication (Anderson. 1998). Recommendation systems mimic these behaviours by utilising the perspectives of many users with similar tastes without the need for them to know each other. This translates into recommendation systems as collaborative filtering.

A recommender system is not limited to just assisting users in making specific choices and has much broader use cases. There are numerous reasons why companies that offer various services will integrate a recommendation system into their offerings (Ricci et al. 2011). From a business standpoint, one of the most important reasons is the chance to boost revenue through customer referrals(Schafer et al. 1999). Another business-oriented objective could be to broaden the range of products sold so as not to rely excessively on a small group of highly sought-after items. This goal is only possible with a recommendation system, as item promotion is risky and costly. Pushing

less well-known products can take time and effort. However, a recommendation system can aid in identifying the appropriate audience for such items.

A well-designed recommendation system has also been shown to improve user satisfaction (Nyugen et al. 2018) and loyalty (Schafer et al. 1999) to a service. Gaining a more profound understanding of user preferences can be beneficial not only for the recommendation system but also for other business operations and promotional activities.

There are various reasons for implementing recommender systems and multiple scenarios in which they can be used. The most common method for recommender systems is making recommendations for movies (Netflix (Bennet et al. 2007), MovieLens (Harper and Konstan 2015) or music (Yahoo! Music (Dror et al. 2011)) or books (Douban Books (Douban 2021)), as data for these types of products are often freely accessible. Recommender systems can also be used in other areas,s such as e-commercial platforms, social job-finding platforms (Paparrizos 2011), and online shopping.

Recommender systems can be implemented for a variety of reasons and in a variety of settings. They are most frequently used to suggest movies, music, and books, as data for these products is often readily available. Examples include Netflix (Bennet et aş. 2007), MovieLens (Harper and Konstan 2015), Yahoo! Music (Dror et al. 2011), and Douban Books (Douban 2021). Additionally, recommendation systems can be used in other industries, such as online retail, job suggestions and online purchasing.

A user's interactions with the system are often referred to as transactions in a recommender system. These transactions can include explicit (Adomavicius, Kwon 2011) and implicit (Jawaheer et al. 2014) feedback. Explicit feedback is when a user explicitly states their opinion or preference, whereas implicit feedback is derived from other actions recorded by the system. An example of this would be if a user plays a song but quickly moves on from it, it can be inferred as a sign of dislike for that song, or when a user does not finish a movie that she started to watch can also be interpreted as a dislike from the systems point of view. It falls on the recommendation system to determine the significance of the signals and how various users may perceive the actions in case of implicit feedback. Alternatively, explicit feedback can be written user reviews or like given to an item.

The construction of a recommendation system can be shaped by the usage patterns of items, including how frequently users interact with an item and how long the item stays relevant. The frequency of transactions can vary depending on the domain. For instance, in a music recommendation system, it is typical for users to receive recommendations for the same songs and artists multiple times. In contrast, in a movie recommendation system, it may not be as suitable to recommend the same film to a user multiple times. Similarly, if a user has beaten a very long video game, the system

should suggest a different game over and over. All these elements should be considered when creating a recommendation system for a specific field(Burke and Ramezani 2011), (Garcin et al. 2013), (Liu et al. 2010).

When creating a recommender system, it is crucial to consider the types of recommendations your system will generate and the context in which the recommendations will be used. Sometimes the main goal might be to accurately predict how users will rate an item when it is presented. This type of problem is called a rating prediction problem (Steck 2013). Alternatively, the objective might be to create a set of the most fitting items catalogue for the user by giving little importance to the predicted ratings' accuracy. Such a case is known as the ranking problem. These two approaches have been discussed in the literature and are the two main types of problems that recommender systems can address.

The rating prediction problem received more attention in the past and has been the primary evaluation measure used in the 2006 Netflix Challenge (Bennet et al. 2007). However, ranking issues have become more important to users and businesses. Ultimately, the ranking of the recommendations is important to the user, not how close the predicted rating is to the user's actual rating. The choice of recommendation task can significantly affect how the recommendation is scored, and it is crucial to keep this in mind when creating the recommender system.

1.1.1 Types of Recommendation Algorithms

There are various ways to classify the algorithms used in recommender systems. A classification system put forward by Burke (Burke 2007) identifies six types of recommender systems: content-based, collaborative filtering, demographic, utility-based, knowledge-based, and hybrid recommender systems.

Content-based recommender systems (Lops et al. 2011), (Pazzani and Billsus 2007) provide suggestions based on the recommended items’ characteristics. These systems analyse the content of the items and use this information to make recommendations to users. For example, suppose a spends more time listening to classical music from the same composer. In that case, the recommendation system is more likely to recommend another music composed by the same composer. This type of recommendation approach can make personalised recommendations to users based on their specific interests. However, one potential limitation is that they may not be able to suggest new items outside of a user's current interests(Lops et al. 2011).

Collaborative filtering recommender systems (Goldberg et al. 1992) provide suggestions depending on the similarities between users. These systems examine the ratings and preferences of multiple users and utilise this data to make suggestions to individual users. Suppose two users have both rated a particular movie highly. In that case, a collaborative filtering recommender system might suggest that movie to one of

the users based on the assumption that they will also enjoy it. One of the benefits of collaborative filtering is that it can make suggestions based on the combined preferences of multiple users, which can help reveal items to users that they may not have found independently. However, one potential limitation is that these systems may need to be more effective for users with unique interests or preferences.

Demographic recommender systems (Ghazanfar et al. 2010) provide suggestions based on the user's demographic characteristics, such as age, gender, or location. These systems may use information about the user's demographic profile to tailor their recommendations to their specific interests or needs. For example, a demographic recommender system might suggest different products or services to a young female user than an older male user. One advantage of demographic recommender systems is that they can make highly personalised user recommendations based on their specific demographic characteristics. However, one potential limitation is that they may not be able to effectively recommend items to users who do not fit into a particular demographic category.

Utility-based recommender systems (Huang and S. L. 2011) provide suggestions based on the utility or value that an item is expected to provide to the user. These systems analyse the utility of different items for a particular user and use this information to make recommendations. For example, a utility-based recommender system might suggest a more expensive but longer-lasting product to a user who values durability over cost. One advantage of utility-based recommender systems is that they can help users make informed decisions by considering the potential long-term value of different items. However, one potential limitation is that they may not be able to effectively recommend items to users who have very different values or priorities.

Knowledge-based recommender systems (Burke and R. 2000) provide suggestions based on the user's knowledge or expertise in a particular domain. These systems may use information about the user's expertise to tailor their recommendations to the user's specific interests or needs. For example, a knowledge-based recommender system might suggest different products or services to a user who is an expert in a particular field than to a user who is less familiar with that domain. One advantage of knowledge-based recommender systems is that they can make highly personalised user recommendations based on their specific knowledge and expertise. However, one potential limitation is that they may not be able to effectively recommend items to users who do not have a particular area of expertise.

Hybrid recommender systems (Adomavicius and Tuzhilin 2005), (Burke 2002) combine two or more of the above approaches to make recommendations. These systems may use a combination of content-based, collaborative filtering, demographic, utility-based, and knowledge-based approaches to make recommendations to users.

1.2 Evaluation Methodologies

It is typical to perform a series of experiments to compare the effectiveness of different algorithms to find out which one works best in a specific scenario or which parameters are optimal for each algorithm. According to Shani and Gunawardana (Shani and Gunawardana 2011), there are three primary methods to evaluate the recommendations generated by these algorithms.

1.2.1 Offline Evaluations

Assessing the performance of recommendation systems using offline evaluation is a common practice. This method uses a dataset of user ratings or choices for specific items. Offline evaluation relies on the assumption that the user's characteristics during the data collection are similar to when using a system trained on this data. The benefit of this method is that it allows for many algorithms to be evaluated efficiently and at a low cost. However, due to the exponential increase in real-time interactions between users and recommendation systems. However, this evaluation does not offer much understanding of the recommendations' quality or the overall effectiveness of recommendation systems in practical scenarios.

When conducting offline evaluation, it is important to use a dataset as similar as possible to the data encountered in the online system. It is important to be cautious to avoid introducing any other biases, as correcting them can be difficult. These biases may arise during the filtering and partitioning process that some recommender systems need or during the data-gathering stage. For instance, with explicit feedback, people might be more inclined to rate items they enjoy, meaning that the data is not missing randomly.

To judge the performance of a recommender system using offline evaluation, it is necessary to have a way to mimic how users would interact with the system is required. A typical approach is to reserve a part of the recorded interactions and use it as a reference or "ground truth." This entails dividing the dataset into a training set, which comprises "known" interactions, and a test set, which comprises "hidden" interactions. There are various methods to do this split, such as randomly dividing the recorded interactions, dividing the interactions based on time or selecting a fixed number of known or hidden items for each user. It is important to consider which technique is most appropriate for evaluating the specific recommendation scenario.

1.2.2 Online Evaluations

In online evaluation, the recommendation system is tested in a real-world setting, where users engage with the system and receive recommendations. Many recommender systems have built-in mechanisms for running online experiments and

comparing the performance of different algorithms. A typical method is to select a random subset of users and direct them to the algorithm being tested while maintaining other system elements unchanged. This grants for a just comparison of the performance of the different algorithms. However, in some cases, it may be necessary to systematically choose a certain group of users to test the algorithm on if it is particularly relevant for that group. It is important to carefully control the conditions of the experiment to ensure that any differences in performance can be attributed to the algorithm being tested rather than other variables.

Before conducting the new recommender systems online evaluation, it is important to consider the potential risks involved. If the algorithm does not perform as expected, it could negatively impact the user experience and potentially harm the business. To mitigate this risk, it is common practice first to perform an offline evaluation of the system to ensure that the results are stable and reasonable. If the results of the offline evaluation are satisfactory, it may be appropriate to proceed with online testing. This approach allows for a more controlled evaluation of the system and can help to minimise the potential negative effects of an unsuccessful experiment.

1.3 Evaluation Metrics

The kind of trouble a recommendation system is applied to is significant because it can influence the choice of algorithms, the format of recommendations provided, and the methods used to evaluate the system's performance.

Rating prediction and ranking. Rating predictions were previously more widely studied, but researchers later found that rankings were more closely aligned with the actual recommendation scenario (Li et al. 2016)

1.3.1 Rating Prediction

In rating prediction, the system's objective is to predict how a user would rate a particular item accurately. This requires the availability of explicit ratings.

1.3.2 Ranking Predictions

A recommender system, in a ranking task, is a tool that helps identify and suggest items that are relevant to a specific user. Unlike rating-based systems, which predict the user's rating for a specific item, the goal of a ranking-based system is to

create a ranked list of items for each user. This list is determined by scores calculated by the recommendation system, which are used to rank the items in descending order. The top-N items, typically chosen from the highest-ranking items, are then suggested to the user. This approach is commonly used in the literature of recommender systems, and it is a valuable tool for personalizing the user's experience and providing more relevant recommendations.

1.4 Challenges In Recommendation

Several challenges to producing recommendations must be addressed to create an effective recommendation system. These challenges include:

Data sparsity: One of the main challenges of producing recommendations is the issue of data sparsity. This refers that, in many cases, users only engage with a small fraction of the items available in a recommendation system. This can make it difficult to generate reliable recommendations for those users, as there is little data on which to base the recommendations.

Cold start problem: Another difficulty is "cold start" trouble, which pertains to the difficulty of providing recommendations for new users or items. This is because there is little or no data available about these entities, making it difficult to generate reliable recommendations.

Personalisation: Personalisation is an important aspect of recommendation systems, as people have different preferences and interests. However, it can be challenging to accurately capture and model individual preferences, especially when limited data is available.

Handling diversity: Recommendation systems should be able to handle a diverse range of items and users, each with its unique characteristics and preferences. This can be challenging, as it requires the recommendation system to handle and model a wide variety of data effectively.

Evaluating recommendations: Another challenge is evaluating the quality of recommendations. This is important to ensure that the generated recommendations are accurate and relevant to the users. Different evaluation metrics can be utilised to measure the effectiveness of recommendations, such as precision, recall, and accuracy.

2. LITERATURE SURVEY

Popularity bias refers to the phenomenon where the algorithm used in the system tends to recommend items that are more popular or have been interacted with more often rather than items that may be more relevant or interesting to the user in recommender systems. This can lead to a lack of diversity in the recommendations and exposure to less popular items, which can be particularly detrimental in cases where the goal is to discover new items.

Researchers have proposed various methods to address the popularity bias problem and increase the diversity of the recommendations without sacrificing too much accuracy. One of the early works in this area is (Jannach et al. 2011), which argues that accuracy is not the only important metric for evaluating the performance of a recommender system and suggests the use of coverage and serendipity as additional metrics.

The use of these additional evaluation metrics helps to address the popularity bias by including the representation of non-popular items in the recommendations. Another important work is (Zhang et al. 2014), which shows that most users have a long-tail distribution of interests, with a few highly preferred items and many items that are of lower preference.

The authors proposed a method to encourage the recommendation of long-tail items to users and showed that their method could increase the diversity of the recommendations without sacrificing too much accuracy. In (Bell 2015), the authors tackled the problem of generating recommendations for new users who do not have any previously-rated items.

They proposed a method incorporating global popularity information to generate more diverse recommendations for new users. This method allows the incorporation of global popularity information to provide new users with a broader range of options to explore. In (Gullo et al. 2016), a solution for the long-tail problem was proposed by adjusting the similarity measure between items used by the recommendation algorithm. By taking into account not only the observed ratings but also the distribution of ratings among the users, the proposed similarity measure increases the diversity of the recommendations.

The method proposed here addresses the popularity bias by focusing on the distribution of ratings among the users. In Gao et al. (2018), a method for addressing the cold-start problem that incorporates trust information from the user's social network is proposed.

Trust information is used to generate more accurate and diverse recommendations for new users by using the trust information from the user's social network to improve the recommendation performance. This method allows

incorporating trust information to overcome the cold start problem, where there is no previous information about the users' preferences. In Yoshida et al. (2020), a two-fold approach that includes both global and local solutions is proposed. The global solutions are item and user clustering, while the local solutions are the re-ranking method, which reorders the items in the recommendation list to promote diversity.

The two-fold approach allows the system to address popularity bias by combining global and local methods, ensuring a balance between accuracy and diversity. Bilge et al., tackle the problem and possible solutions for the popularity bias problem in search engines. They proposed using diversity metrics to evaluate the performance of the search engines, which allows them to address the popularity bias by providing more diverse results. Overall, the literature on popularity bias in recommender systems suggests a trade-off between accuracy and diversity in recommendations. Many different methods have been proposed to address this problem, from adjusting evaluation metrics, encouraging recommendation of long-tail items, incorporating global popularity information, adjusting similarity measures, incorporating trust information, and using re-ranking methods. These methods help to address the popularity bias by providing a more diverse set of recommendations without sacrificing too much accuracy.

3. MATERIALS AND METHODS

The Materials and Methods section of this thesis presents the research design, techniques, and procedures used to conduct the study. It includes a comprehensive description of the materials, equipment, and software used, as well as the methods for data collection, analysis, and interpretation that were employed to obtain the results.

3.1 User-Based Collaborative Recommendation

In commercial recommendation systems, the Collaborative Filtering recommendation algorithm is a popular individualised recommendation method (Adomavicius et al. 2005). An algorithm called collaborative filtering is based on the following ideas: Humans tend to share preferences and interests closer to each other. Since these traits are stable, we can forecast their choice contingent on previous desires. Owing to the presumptions, a collaborative filtering algorithm compares a user's activity with that of other users to identify his closest neighbours and predicts his interests and preferences based on those of his neighbours.

The user profile history, which may be seen as a rating matrix with all rows representing score a user gave to an item, is obtained as the first phase of the collaborative filtering method. In a rating matrix, each row in the table corresponds to a user, each column to a particular item, and the number at the intersection of each row and column to the user's rating value. If there is no rating score at this junction, the user has not yet given the item a rating. Due to the issue with sparse scoring, we substitute a list for a matrix.

6 The similarity between users is determined in the second stage, followed by the location of nearest neighbours. There are several ways to measure similarity. In our study, we have used three similarity metrics to be discussed in the upcoming sections, Cosine similarity, Pearson’s correlation coefficient similarity, and Mean Squared Difference similarity.

3.2 Popularity Bias in Recommender Systems

The proliferation of digital platforms has led to a significant increase in the number of products and services available on the Internet, which can complicate the decision-making process of individuals. To help users find desired products and services, the referral system has been widely implemented on various platforms, such as Booking.com, Amazon, YouTube, and Spotify. These systems typically use collaborative, content-based, or hybrid filtering to predict the ratings of items that have not been tried or recommend a list of items prioritised for ratings. While these recommendation algorithms can often make accurate recommendations, they suffer from a well-known problem called popularity bias. Certain popular items are suggested

too often when niche items are less suggested in the proposals, even attractive. This can reduce the quality of proposals, especially in terms of diversity, category coverage, and novelty. Furthermore, common trends can be exploited by malicious users or vendors to increase sales or even bring down the entire system. Additionally, this bias can leave individuals entrenched in feedback rooms and potentially harm social ethics, especially in news and social media.

3.2.1 Pareto Principle

To comprehend popularity bias, we have separated the items in the catalogue into two groups, a head and a tail list. This process was done by utilising the Pareto Principle (Sanders 1987). The items that have received the most ratings, which account for 20% of all ratings, are referred to as the head, while the remaining items in the catalogue are referred to as the tail. This principle is applied throughout all of our tests.

3.3 K Nearest Neighborhood Recommendation Algorithms

K-nearest neighbors (KNN), is a method of making recommendations that is built upon the idea of finding similar items. Given a dataset, a set of items and a target item, the algorithm finds the k number of items in the dataset that are most similar to the target item and recommends those items to the user. The similarity is usually determined by calculating the distance between the feature vectors of the items using a distance metric such as Euclidean distance. KNN is a simple but powerful algorithm that can be used for both classification and regression tasks and is often used in collaborative filtering and content-based recommendation systems.

3.3.1 KNN With Means

KNNWithMeans is a method for making recommendations through collaborative filtering, where it considers the ratings of similar users to make suggestions using the k-nearest neighbours algorithm. It is an extension of the basic KNN algorithm that takes the mean ratings for all users when making recommendations.

The KNNWithMeans algorithm first calculates the similarity between users based on their ratings for a given set of items (3.3.1). The similarity between users is usually measured using a distance measure, such as cosine similarity or Pearson's correlation coefficient. Once the similarity between users is calculated, the algorithm selects the k users most similar to the target user and uses their ratings to predict the target user.

The main improvement of the KNNWithMeans algorithm is to use the average rating of each user as a reference when making predictions. By taking into account the average rating of each user, the algorithm can make more accurate predictions taking into account that different users may have different overall preferences for items.

KNNWithMeans has been widely studied in the recommendation systems literature and is an effective approach to making personalised recommendations. (Adomavicius et al 2005), (Cheng et al. 2016). It is particularly useful for scenarios with limited user-item interaction data available, as it can effectively incorporate the ratings of similar users to make predictions.

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u,v)} \quad (3.3.1)$$

In the equation (3.3.1), u and v represent two different users, while i and j represent two different items. r_{ui} Represents the score that the user u gave to item i and r_{vi} Represents the score that the user v gave to item i. And μ_v Is the average rating given by the user u to all items. $\hat{r}_{ui}$ Is the recommender system guessed score for the rating that user u gave to item i. k represents the maximum number of neighbours to take into account for aggregation and $N_i^k(u)$ is the set of k most similar users to user u who have rated item i. Finally, the $sim(u,v)$ Represents the similarity between user u and v.

3.3.2 KNNBaseline

KNNBaseline is a collaborative filtering algorithm that uses k-nearest neighbors (KNN) to make recommendations based on the ratings of similar users. It is an extension of the basic KNN algorithm that includes a baseline prediction for each user to improve the recommendations' accuracy.

The KNNBaseline algorithm first calculates the similarity between users based on their ratings for a given set of items (3.3.2). The similarity between users is usually measured using a distance measure, such as cosine similarity or Pearson's correlation coefficient. Once the similarity between users is calculated, the algorithm selects the k users most similar to the target user and uses their ratings to predict the target user.

The key innovation of the KNNBaseline algorithm is using a baseline prediction for each user. The baseline prediction is calculated as the average rating for a given item across all users. It is used as a reference point to adjust the predictions made by the

KNN algorithm. By incorporating the baseline prediction, the KNNBaseline algorithm can make more accurate recommendations by accounting for an item's overall popularity and the target user's individual preferences.

KNNBaseline has been widely studied in the recommendation systems literature and is an effective approach for making personalised recommendations (Koren et al. 2009), (Bobadilla et al. 2013). It is particularly useful for scenarios with a high level of noise in the user-item interaction data, as the baseline prediction helps filter out the noise and improve the accuracy of the recommendations.

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)} \quad (3.3.2)$$

In the equation (3.3.2), u and v represent two different users, while i represents the item. b_{ui} represents the baseline estimate of rating for user u and item i while b_{vi} represents the baseline estimate of rating for user v and item i and r_{vi} Represents the score that the user v gave to item i .$\hat{r}_{ui}$ Is the recommender system guessed score for the rating that user u gave to item i. k represents the maximum number of neighbours to take into account for aggregation and $N_i^k(u)$ is the set of k most similar users to user u who have rated item i. Finally, the $sim(u,v)$ Represents the similarity between user u and v.

3.3.3. KNNWith Z Score

KNN W KNNWithZScore is a collaborative filtering algorithm that uses k-nearest neighbours (KNN) to make recommendations based on the ratings of similar users. It is an extension of the basic KNN algorithm that normalises the ratings of each user using a z-score transformation to improve the recommendations' accuracy.

The KNNWithZScore algorithm first calculates the similarity between users based on their ratings for a given set of items (3.3.3). The similarity between users is typically calculated using a distance measure, such as cosine similarity or Pearson’s correlation coefficient. Once the similarity between users has been calculated, the algorithm selects the k users most similar to the target user and uses their ratings to predict the target user.

The key innovation of the KNNWithZScore algorithm is using a z-score transformation to normalise the ratings of each user. The z-score transformation is a statistical method that standardises the ratings of each user by subtracting the mean rating and dividing by the standard deviation. By normalising the ratings this way, the KNNWithZScore algorithm can make more accurate recommendations by accounting for the individual variability in rating patterns between users.

KNNWithZScore has been widely studied in the recommendation systems literature and is an effective approach for making personalised recommendations (Cantador et al. 2011), (Dacrema et al. 2019)

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)} \quad (3.3.3)$$

In the equation (3.3.3) $\hat{r}_{ui}$ Is the predicted rating that user u would give to item i while r_{vi} Is the rating that user v gave to item i and μ_v Is the average rating given by the user u to all items. σ_u stands for deviation of the ratings given by user u to all items, while σ_v Stands for deviation of the ratings given by user v to all items. $N_i^k(u)$ is the set of k most similar users to user u who have rated item i. Finally, $sim(u,v)$ Stands for the similarity between user u and v.

3.4 Similarity Metrics

3.4.1 Cosine Similarity

Cosine similarity is a method that calculates the similarity between two vectors, and it is often used in recommender systems to determine the similarity between users or items. Cosine similarity is determined by measuring the cosine of the angle between two vectors. This is found by dividing the vectors' dot product by the product of the vectors' magnitudes. (3.4.1).

In recommender systems, cosine similarity is often used to calculate similarity among users based on their ratings for a given set of items. To calculate the similarity between two users, each user's rating is treated as a vector, and the cosine similarity between the vectors is calculated. Similarity score results range from -1 to 1, where a score of 1 indicates users with identical ratings, and a score of -1 indicates users with opposite ratings.

Cosine similarity is a popular choice for recommendation systems because it is easy to calculate and does not require the vectors to be normalised. It is also robust to

the presence of missing values in the rating data, as the missing values can be treated as zero in the calculation.

Cosine similarity has been widely studied in the recommendation systems literature and has been shown to be an effective measure of similarity for making personalised recommendations (Liang et al. 2016), (Sarwar et al. 2001) It is particularly useful for scenarios where the rating data is sparse, as it is able to effectively capture the similarity between users even when they have rated only a small number of items.

$$cosine_{(u,v)} = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}} \quad (3.4.1)$$

In the equation (3.4.1), u and v are two different users. r_{ui} Is the rating that user u gave to item i and r_{vi} is the rating that user v gave to item i. I_{uv} Stands for the set of items that both user u and v rated.

3.4.2 Pearson's Correlation Coefficient

Pearson's correlation coefficient (Pearson) similarity is a measure of similarity between two vectors commonly used in recommender systems to calculate the similarity between users or items. It is based on Pearson's correlation coefficient, which measures the linear relationship between two variables.

In recommender systems, Pearson similarity is often used to calculate similarity among users based on their ratings for a given set of items. To calculate the similarity between two users, each user's rating is treated as a vector, and the Pearson's correlation coefficient between the vectors is calculated (3.4.2). The obtained similarity scores range from -1 to 1, where 1 indicates a perfect positive correlation between user ratings and a score of -1 indicates a completely negative correlation.

Pearson similarity is a popular choice for recommendation systems because it can capture the strength and direction of the relationship between the users' ratings. It can also handle missing values in the rating data, as the correlation coefficient is calculated using the observed ratings and is not affected by missing values.

However, Pearson similarity has some limitations in recommendation systems. It is sensitive to outliers in the rating data and can influence each user's overall preference level. It is also less effective at capturing the similarity between users with very different rating scales.

Pearson similarity has been widely studied in the recommendation systems literature and is an effective measure of similarity for making personalised recommendations (Pilászy et al. 2010), (Shi et al. 2014). It is particularly useful for scenarios where the rating data is dense, and the relationship between the users' ratings is linear.

$$pearson_{(u,v)} = \frac{\sum_{i \in I_{uv}} (r_{ui} - \mu_u) \cdot (r_{vi} - \mu_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \cdot \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}} \quad (3.4.2)$$

In the equation (3.4.2) u and v are two different users. r_{ui} is the rating that user u gave to item i and r_{vi} is the rating that user v gave to item i. While μ_u is the mean of all ratings that the user u gave and μ_v is the mean of all ratings that the user v gave. And I_{uv} stands for set of items rated by both users u and v.

3.4.3 Mean Squared Difference

MSD similarity, also known as mean squared difference (MSD) similarity, is a measure of similarity between two vectors that are commonly used in recommendation systems to calculate the similarity between users or items. It is based on the mean squared difference between the ratings of the two vectors.

In recommendation systems, MSD similarity is often used to calculate the similarity between users based on their ratings for a given set of items. To calculate the similarity between two users, the ratings of each user are treated as a vector, and the mean squared difference between the vectors is calculated. The resulting similarity score is a non-negative value, where a smaller score indicates a higher level of similarity between the users.

MSD similarity is a popular choice for recommendation systems because it can capture the overall difference in the users' ratings, including both the magnitude and direction of the difference. It is also relatively easy to calculate and does not require normalising the vectors.

However, MSD similarity has some limitations in recommendation systems. It is sensitive to outliers in the rating data and can be influenced by each user's overall level of preference. It is also less effective at capturing similarities between users with very different rating scales.

MSD similarity has been widely studied in the recommendation systems literature and has been shown to be an effective measure of similarity for making personalised recommendations (Herlocker et al. 2004). It is particularly useful for scenarios where the rating data is dense, and the relationship between the users' ratings is linear.

$$msd_{sim(u,v)} = \frac{1}{\frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2 + 1} \quad (3.4.3.1)$$

In the equation (3.4.3.1) u and v are two different users. r_{ui} is the rating that user u gave to item i and r_{vi} is the rating that user v gave to item i. And I_{uv} stands for set of items rated by both users u and v.

3.5 Datasets

3.5.1 MovieLens-1M

The MovieLens dataset is a collection of data made available by the GroupLens, which is based at the University of Minnesota; MovieLens dataset is a collection of movie ratings and demographic data from the MovieLens website. It is widely used in recommendation systems research to test algorithms' performance and study other related topics (Harper and Konstan 2016), (Steck et al. 1995). The MovieLens 1 million dataset (MLM) contains approximately 1 million and 40 ratings from 6,040 users for 3,952 movies. This means that the dataset includes ratings from approximately 1,666 users for each movie in the dataset. The MLM dataset is suitable for studying popularity bias due to its size, diversity, and the inclusion of a range of ratings, as well as its widespread use in previous research on recommendation algorithms.

3.5.2 Yahoo!Music

The Yahoo!Music (YM) dataset is a set of users listening to data from the Yahoo! Music service, which was one of the first websites to offer DRM-free (media without digital rights management) downloads. It includes information about the tracks listened to by users and metadata about the tracks, artists, and albums. The dataset has been used to test recommendation algorithms' performance and study other related topics. The Yahoo!Music dataset is suitable for studying popularity bias due to its size,

diversity, and the inclusion of various types of listening events and its use in previous research on recommendation algorithms (Herlocker et al. 2004), (Konstan et al. 1997).

3.5.3 Douban Book

The Douban Book (DB) dataset is a collection of book ratings and reviews provided by users of the Douban website, a social networking service in China (Reference: Douban, 2021). It has been widely used in research on recommendation systems and has been the subject of numerous papers and articles (Douban 2021), (Li et al. 2010).

One reason that the Douban Book dataset is an important dataset for testing popularity bias is that it contains ratings and reviews from a large number of users, which allows researchers to examine how recommendations made by a recommendation system are affected by the popularity of items in the dataset. Additionally, the Douban Book dataset includes a wide variety of books, which allows researchers to evaluate the performance of a recommendation system on items with different levels of popularity.

Overall, the Douban Book dataset is a valuable resource for researchers studying recommendation systems and the impact of popularity on recommendations. It is a rich data source that can be used to better understand the factors that influence recommendation algorithms and identify ways to improve their performance (Chen et al. 2012), (Zhou et al. 2014).

More comprehensive details about datasets are given in Table 3.5.3.

Table 3.5.3. Statistics of MovieLens-1M, Douban Book and Yahoo!Music, where U is the set of users, I is the set of items, R is the set of all provided ratings, Avg_R is the average of available ratings, and sparsity is defined as the ratio of unobserved ratings, i.e., $|U|\ X\ |I| - |R|$, to the number of possible all ratings $|U|\ X\ |I|$.

Dataset	**Domain**	$\|U\|$	$\|I\|$	$\|R\|$	$Sparsity(\%)$	Avg_R	$\frac{\|R\|}{\|I\|}$	$\frac{\|R\|}{\|U\|}$
MovieLens-1M	Movie	6,040	3,952	1,000,209	95.8	3.58	253.1	165.6
Yahoo!Music	Music	15,400	1,000	311,704	97.9	2.89	311.8	20.2
Douban Book (DB)	Book	13,024	22,374	792,062	99.7	4.05	35.4	60.8

3.6 Evaluation Metrics for Popularity Bias

The chosen metrics for our experiments are the Average Percentage of Popular Recommended items (APRI), Average Percentage of Long Tail Items (APLT), Long Tail Coverage (LTC) and Entropy.

3.6.1 Average Percentage of Popular Recommended Items (APRI)

The Average Percentage of Popular Recommended items (APRI) measures the effectiveness of a recommendation system. It is calculated as the percentage of items recommended by the system that is also popular, where the popularity of an item is defined as the percentage of users who have interacted with the item. (3.6.1)

APRI is often used as an evaluation metric for recommendation systems because it helps to ensure that the recommendations made by the system are not only relevant to the user but also widely liked by other users. This can help to increase the likelihood that the user will find the recommendations valuable and engaging (Rendle et al. 2009), (Lu et al. 2012).

$$APRI_u = \frac{\sum_{i \in N} P_i}{|N_u|} \tag{3.6.1}$$

In the equation (3.6.1) P_i stands for the popularity of item i and N_u Represents the top-N item list of user u.

3.6.2 Average Percentage of Long Tail Items (APLT)

The Average Percentage of Long Tail Items (APLT) is a measure of the effectiveness of a recommendation system in recommending items from the "long tail" of the item distribution. The long tail refers to the large number of infrequently interacted items in a dataset, as opposed to the small number of highly popular items that make up the "head" of the distribution.

APLT is calculated as the percentage of recommended items that belong to the long tail of the item distribution (3.6.2). A high APLT indicates that the recommendation system can recommend a diverse set of items, including those that are less popular. This can be beneficial because it can help to prevent the recommendation system from showing a bias towards popular items, also known as the "popularity bias."

Popularity bias is a common issue in recommendation systems. The system tends to recommend popular items more often, even if they may not be as relevant or

interesting to the user as other less popular items. This can lead to a lack of diversity in the recommendations and may result in a less engaging user experience.

APLT is a useful metric for evaluating the performance of a recommendation system in terms of its ability to overcome popularity bias and recommend a diverse set of items (Lu et al 2015), (Li et al. 2016).

$$APLT_u = \frac{|\{i, i \in (N_u \cap T)\}|}{|N_u|} \quad (3.6.2)$$

In the equation (3.6.2) T represents set of tail items in the catalogue of items and N_u represents the items that are recommended in user u's top-N list.

3.6.3 Long Tail Coverage (LTC)

Long Tail Coverage (LTC) is a measure of the effectiveness of a recommendation system in recommending items from the "long tail" of the item distribution. The long tail refers to a large number of infrequently interacted items in a dataset, as opposed to the small number of highly popular items that make up the "head" of the distribution.

LTC is calculated as the percentage of unique long-tail items recommended by the system over a given period (3.6.3). A high LTC indicates that the recommendation system can recommend a diverse set of long-tail items rather than just the most popular items. This can be beneficial because it can help to prevent the recommendation system from showing a bias towards popular items, also known as the "popularity bias."

LTC is a useful metric for evaluating the performance of a recommendation system in terms of its ability to overcome popularity bias and recommend a diverse set of long-tail items (Zhang et al. 2018), (Saha et al. 2018).

$$LTC = \frac{|\bigcup_{u \in U}(l_u \cap (M \cup T))|}{|M \cup T|} \quad (3.6.3)$$

In the equation (3.6.3) M represents the head items while T represents the tail items. l_u represents the set of items consumed by user u.

3.6.4 Entropy

Entropy, in information theory, measures the amount of uncertainty or randomness in a system. It is often used to quantify a dataset's diversity or variety of items.

Regarding the use of recommendation systems, Entropy can measure the diversity of an individual's preferences, or the distribution of ratings provided by a group of users. A high entropy value indicates that the preferences or ratings are distributed evenly across a wide range of items. In contrast, a low entropy value indicates that the preferences or ratings are concentrated on a small number of items.

Entropy can be useful for determining the effectiveness of a recommendation system regarding its ability to overcome popularity bias. By measuring the Entropy of the recommendations made by the system, it is feasible to evaluate the system's capability to provide suggestions. a diverse set of items or if it is biased towards popular items (Shannon and C.E. 1948), (Lu et al. 2012).

There are several ways to calculate Entropy, but a common approach is to use the following formula (3.6.4):

$$\varepsilon_u = -\sum_{i=1}^{k} P(r_i) \log_2(P(r_i)) \tag{3.6.4}$$

where $P(r_i)$ is the probability of item i and the summation is over all items in the dataset.

3.7 Experimental Setup

To investigate the impact of different recommendation algorithms on popularity bias, we conducted an experiment using three datasets: MLM, YM, and DB. The experiment aimed to determine which algorithms and parameter configurations were most effective at reducing popularity bias and recommending relevant items.

To experiment, we ran three KNN recommendation algorithms on each dataset: KNNBaseline, KNN with Means, and KNN with Z Score. KNN algorithms are a popular choice for recommendation systems because they can effectively capture the similarity between items based on user interactions. By running multiple algorithms, we could compare the performance of different approaches and see how they impacted recommendation quality.

We varied the experiment conditions by running each algorithm with three different values for the number of nearest neighbours (k) and three different similarity metrics (Cosine, Pearson, and MSD). This allowed us to test various algorithms and parameter configurations and see how they impacted recommendation performance. By using multiple values for k and different similarity metrics, we explored the algorithms' sensitivity to these parameters and determined which configurations were most effective at reducing popularity bias.

After running the tests, we collected the results for four metrics: APLT, APRI, LTC, and Entropy. These metrics allowed us to evaluate the performance of the algorithms regarding their ability to recommend relevant items and their impact on popularity bias. APLT, Novelty, LTC, and Entropy measure the novelty and diversity of the recommendations. By collecting results for these metrics, we were able to get a comprehensive understanding of the strengths and weaknesses of the different algorithms with different test parameters.

To analyse the results, we used statistical techniques to evaluate the effectiveness of the algorithms by comparing them across the different datasets and parameter configurations.

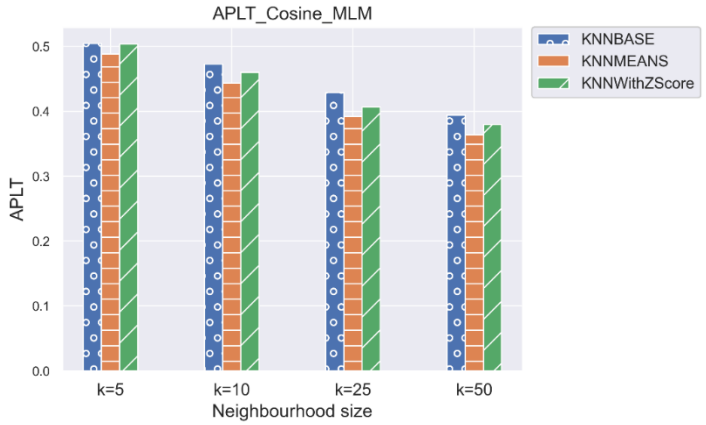
Overall, our experiment demonstrated the importance of carefully considering the choice of recommendation algorithm and parameter configurations when designing a recommendation system. By carefully selecting the right algorithm and parameters, we can significantly enhance the excellence of the suggestions made by the system. and reduce the impact of popularity bias on the system.

In the configuration of the algorithms used for providing recommendations, the top 10 recommendations list for each user has been collected.

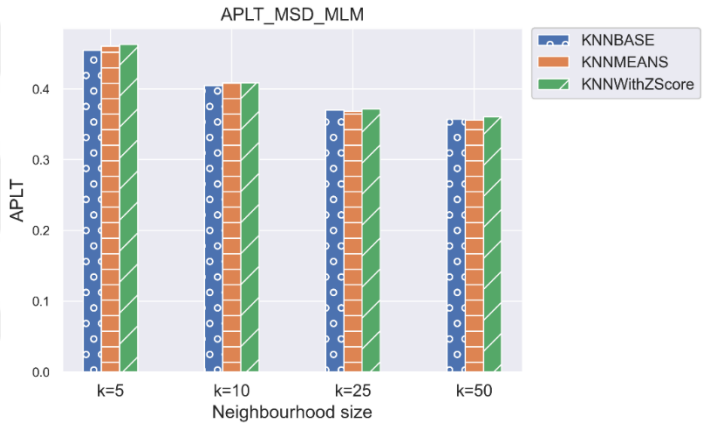
4. RESULTS and DISCUSSION

This section will provide our test results and comment on each metric tested in this study comprehensively.

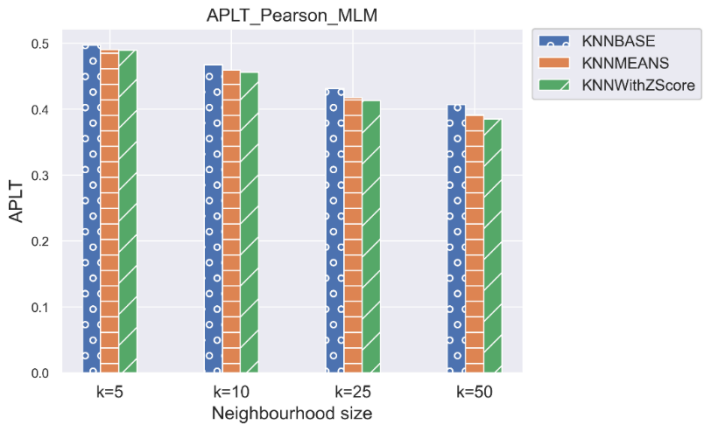
4.1 APLT Results



(a)



(b)

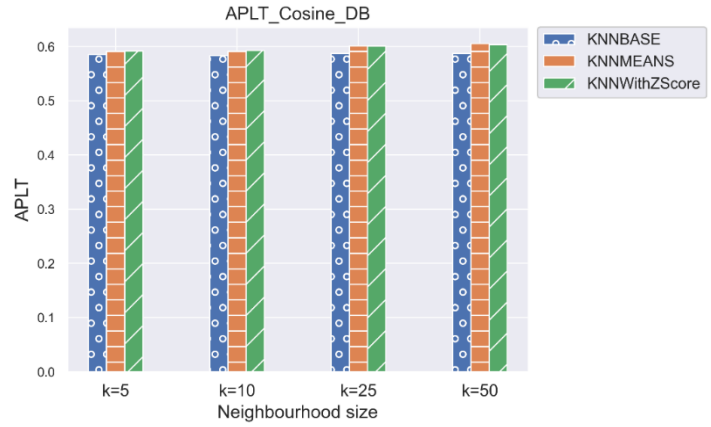


(c)

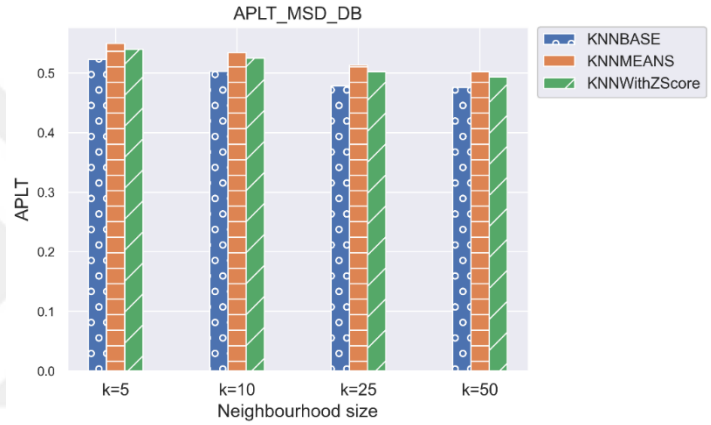
Figure 4.1 APLT Results for MovieLens-1M

Figure 4.1 for the MLM dataset shows that with increasing neighbourhood sizes, all algorithms and similarity metrics show that the APLT metric decreases significantly. This shows that an increase in a neighbourhood creates more popularity bias. When we dive into the results, it can be seen that using cosine similarity metrics APLT metric is highest with KNNBaseline and lowest in KNNWithMeans algorithms in all cases. In MSD similarity ranking of algorithms in terms of APLT metric changes, KnnWithZscore takes the lead while KNNBaseline is the lowest. However, in Pearson similarity, KNNBaseline becomes the algorithm with the highest APLT metric, while KNNWithZScore is the lowest. In Cosine and Pearson similarities, APLT values are higher compared to MSD similarity.

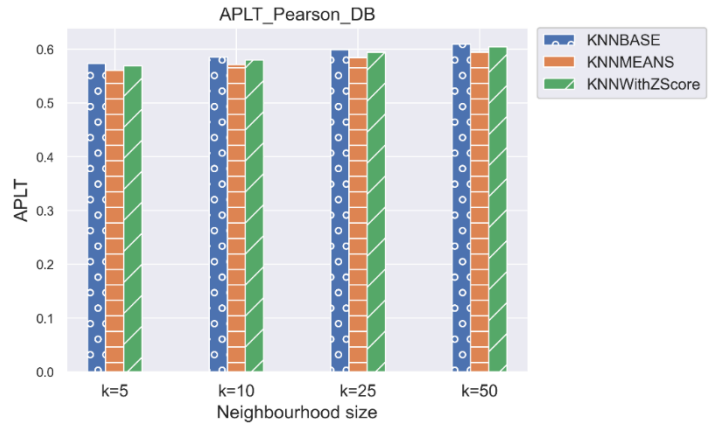
It can be seen from the results that in a dense dataset such as MLM, reducing the neighbourhood sizes helps fight popularity bias. However, using MSD similarity as the similarity technique, it can be seen that lover numbers of neighbourhood size help fight popularity bias; MSD seems to perform worse than other similarity metrics in dealing with popularity bias. While in terms of different recommendation algorithms, KNNBaseline is better at dealing with popularity bias in a dense dataset such as MLM.



(a)



(b)

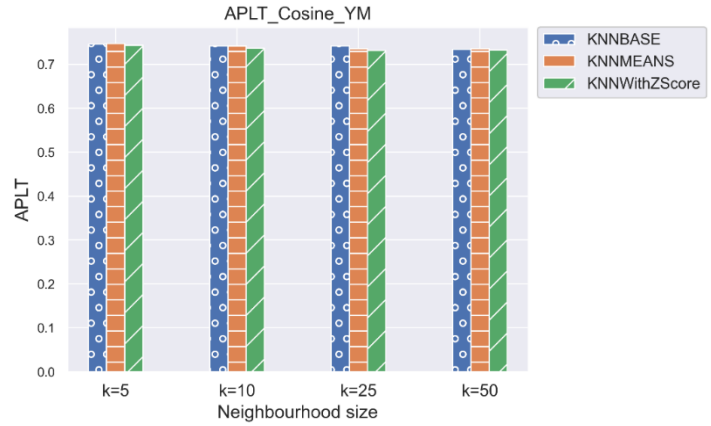


(c)

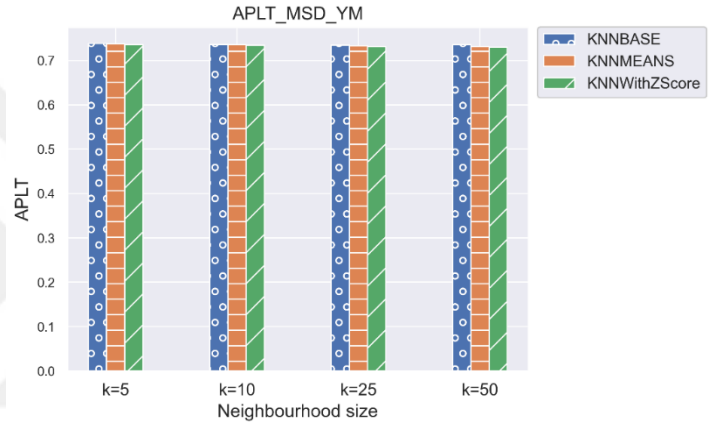
Figure 4.2 APLT Results for Douban Books

In Figure 4.2 it can be seen that with the increasing neighbourhood sizes, the APLT metric is not affected as much as the case with the MLM dataset. In this case, comparing metrics between algorithms, it can be seen that in cosine similarity and MSD similarity, KNNWithMeans takes the lead, continued by KNNWithZScore, however in Pearson similarity, the KNNBaseline algorithm becomes the one with the most APLT score in all tests, continued with KNNWithZScore. With Cosine and Pearson similarities, APLT values are close to each other, while in MSD, there is a significant fall in the APLT.

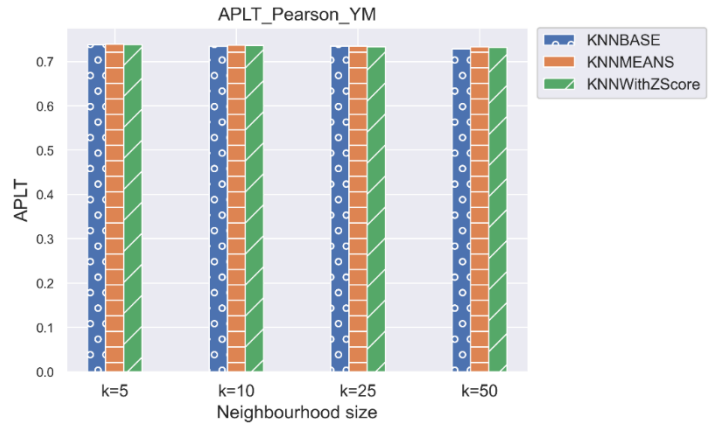
Compared to the MLM dataset, the DB dataset somewhat does not get effected as much as MLM from neighbourhood size in terms of APLT. The cause is that DB is a much sparse dataset compared to MLM. It can be said that sparse datasets are not affected as much by neighbourhood sizes compared to dense datasets. However, they get more effected than the chosen similarity metrics. Even though MLM is affected by similarity metrics, in DB, similarity metrics can change the effect of neighbourhood sizes. Where can be seen in Figure 3. c unexpectedly with Pearson similarity neighbourhood size helps to fight popularity bias.



(a)



(b)



(c)

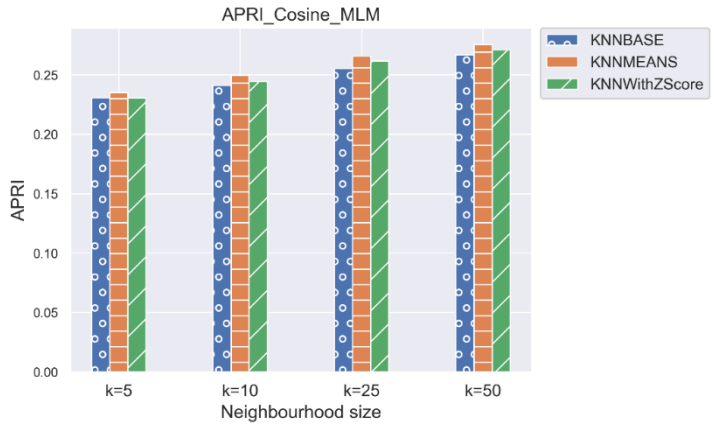
Figure 4.3 APLT Results for Yahoo!Music

In Figure 4.3 it is seen that with the YM dataset, the results are neither affected by similar metrics nor neighbourhood size.

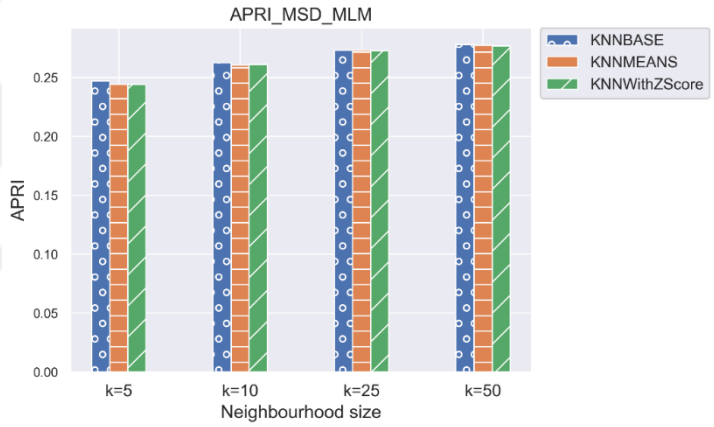
Since YM is a much sparse dataset, there is a small popularity bias compared to other datasets with the same tests applied. The ratings are too low compared to other chosen datasets in the YM dataset. The music domain itself seems to generate less bias overall compared to the books or movies domain.

To generally speak about the APLT metric in our conducted test, it can be said that it is possible to reduce popularity bias by adjusting recommendation algorithms variables without defining a new algorithm to target dealing with popularity bias specifically. For dense datasets, it is clear that with the increasing numbers of neighbours, regardless of the chosen similarity metric, there is always an increase in popularity bias. The KNNBase method seems to perform better than other algorithms. However, in a sparse dataset such as DB, it is better not to use MSD and increase the neighbourhood sizes to deal with the popularity bias.

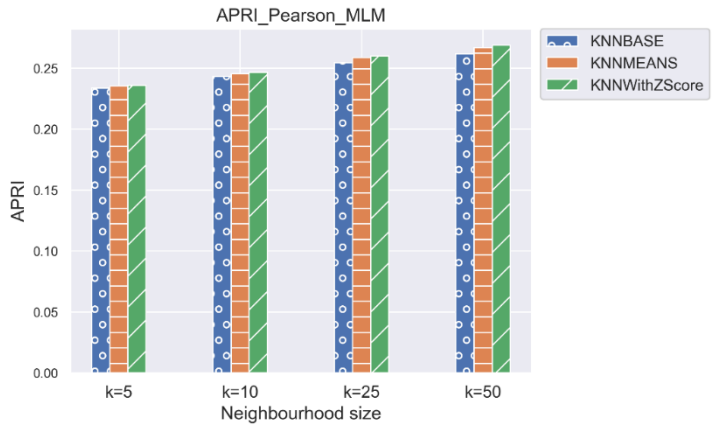
4.2 APRI Results



(a)



(b)

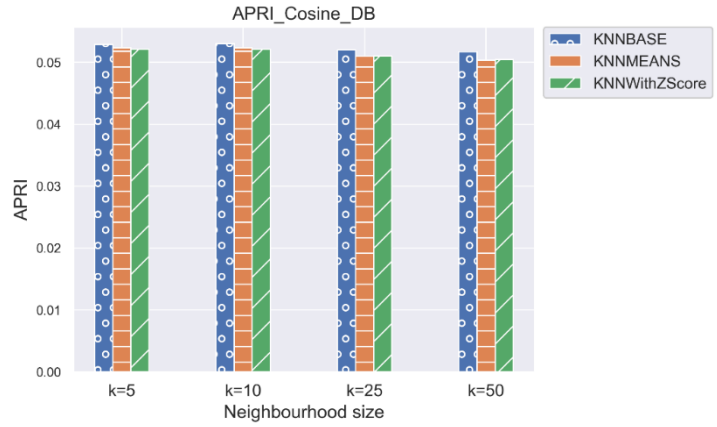


(c)

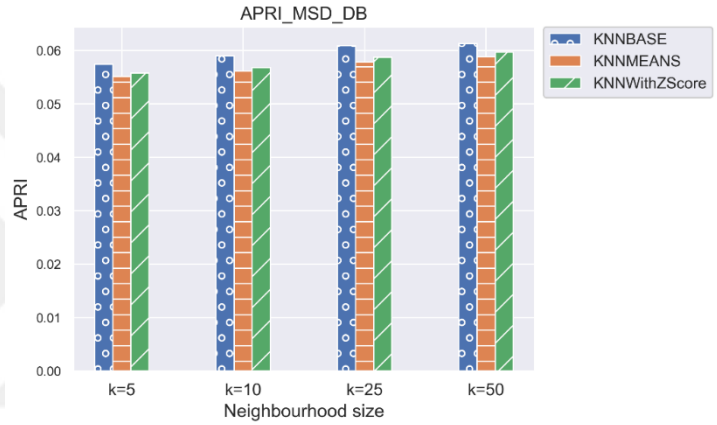
Figure 4.4 APRI Results for MovieLens-1M

Figure 4.4 shows a positive correlation between the APRI metric and neighbourhood sizes for all test cases with the MLM dataset. Again, the similarity metrics play roles in the APRI values compared between each different algorithm tested. In the Cosine similarity, KNNWithMeans has the highest APRI values, followed by KNNWithZScore. With the MSD similarity, it is seen that algorithms perform closer to each other compared to other similarity metrics. KNNBaseline takes the lead in APRI values, followed very closely by KNNWithZScore. Generally speaking, the APRI values between different similarities are not affected significantly.

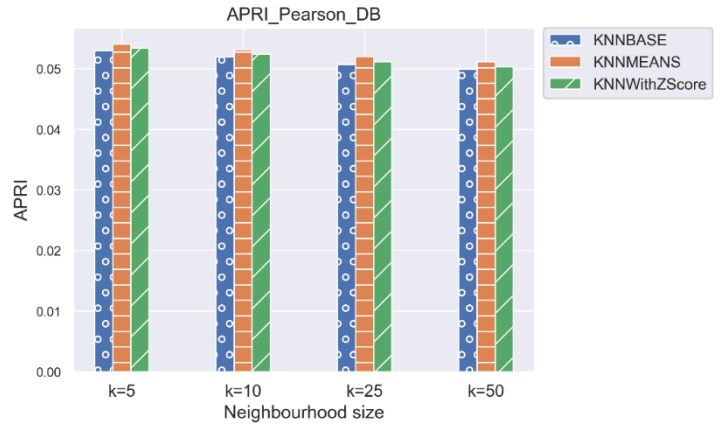
With the increase in neighbourhood size, the popularity bias increases in both test cases with the MLM dataset. This is expected since, by considering more neighbours, the recommendation algorithm comes across more head items. Also, it is observed that similarity metrics do not have different effects compared to each other. It should also be noted that KNNBaseline again seems to deal with popularity bias comparatively better than other algorithms.



(a)



(b)

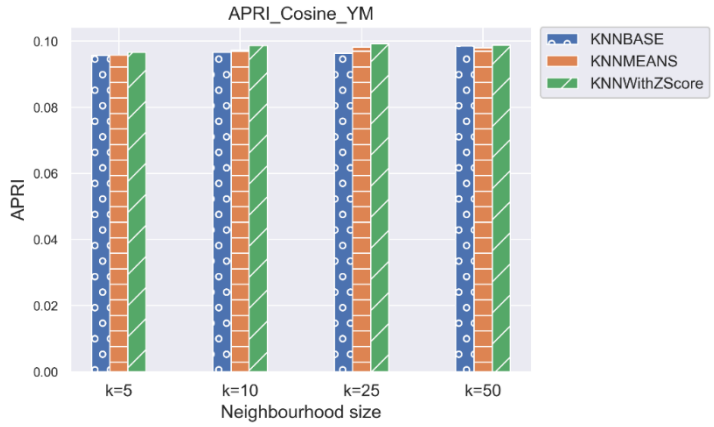


(c)

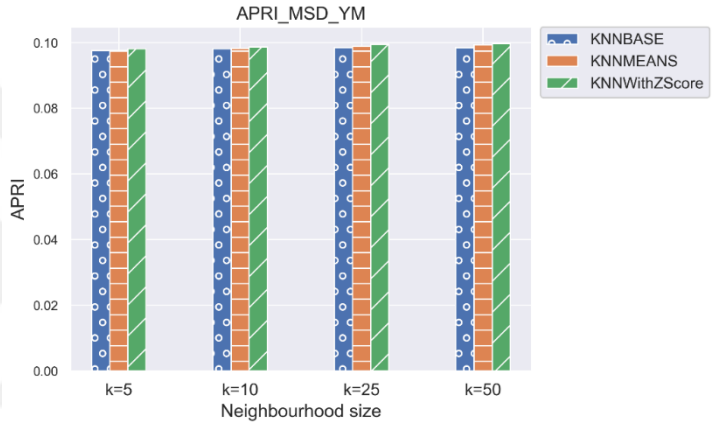
Figure 4.5 APRI Results for Douban Books

It is seen in Figure 4.5 that with the DB dataset, there is a negative correlation between APRI and neighbourhood sizes in Cosine and Pearson similarity metrics; however, with MSD similarity, this correlation becomes positive, while between all similarity metrics, APRI values stay close to each other between different cases. With different similarity metrics, it can also be seen that when different recommendation algorithms are compared, ranking between them in terms of APRI values changes.

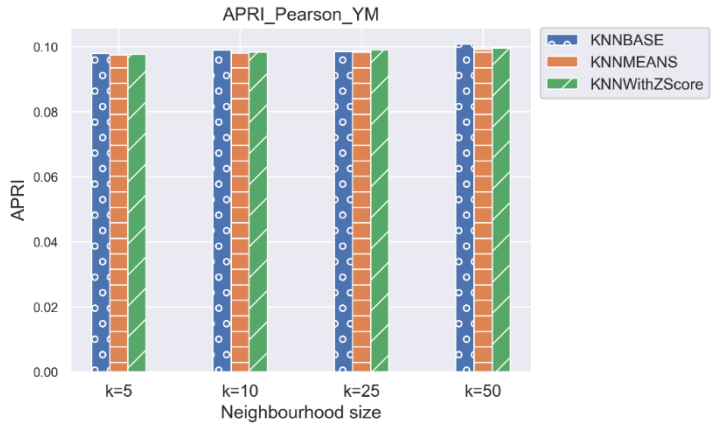
It can be noted that APRI values are close to each other in all test cases. Again, neighbourhood sizes or different similarity metrics can not significantly affect the result in a sparse dataset.



(a)



(b)



(c)

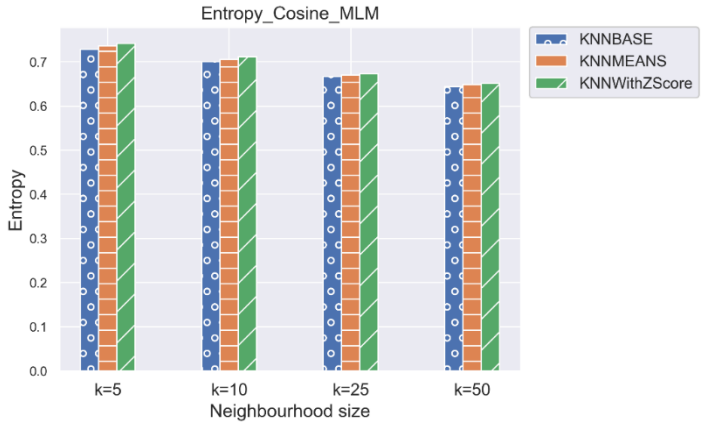
Figure 4.6 APRI Results for Yahoo!Music

In Figure 4.6 it is seen in the YM dataset that there can be a small amount of positive correlation between neighbourhood size and APRI values in cosine similarity; however, in terms of other similarity metrics, there cannot be any significant change observed.

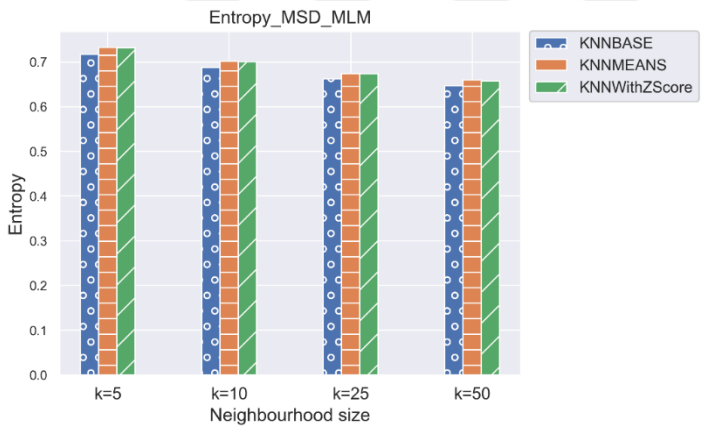
In the most sparse dataset we have tested (YM), it can be seen that the APRI is not affected by our different test scenarios.

Overall, in APRI, it can be seen that working with a sparse dataset, there cannot be any significant effect from the metrics and scenarios used in our experiments.

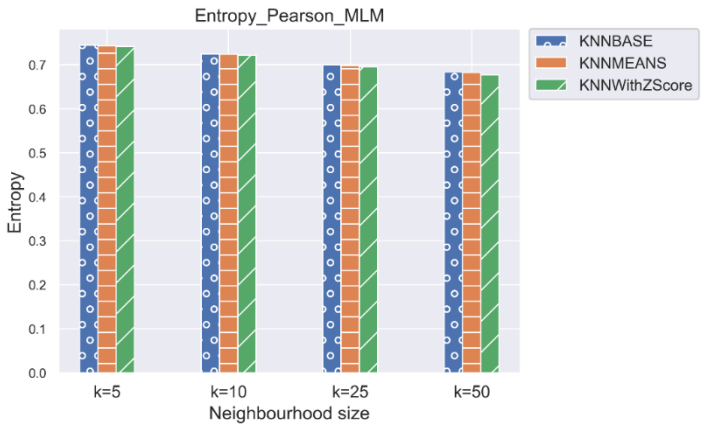
4.3 Entropy Results



(a)



(b)

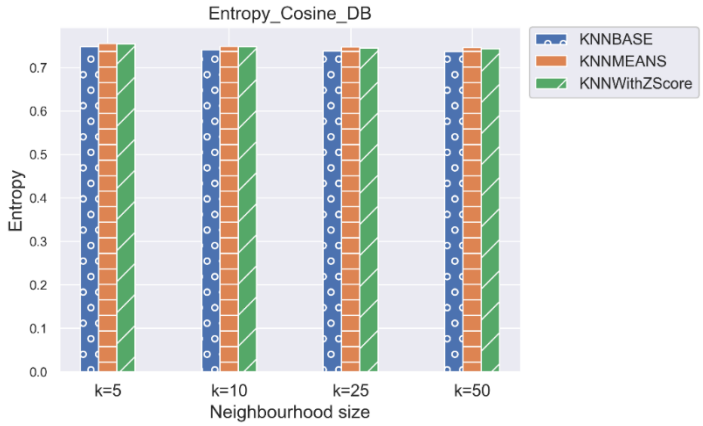


(c)

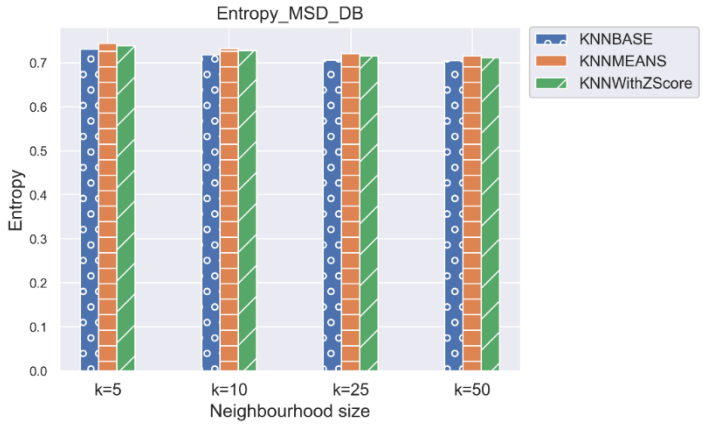
Figure 4.7 Entropy results for MovieLens-1M

In all different similarity metrics, a negative correlation between neighbourhood size and Entropy value can be observed in Figure 4.7 Changes between the neighbourhood size and Entropy seem to be more significant in Cosine similarity compared to MSD similarity and Pearson similarity. However, Entropy values do not vary significantly between different similarity metrics. Again, it can be seen that different similarity metrics change the ranking between different recommendation algorithms' Entropy values.

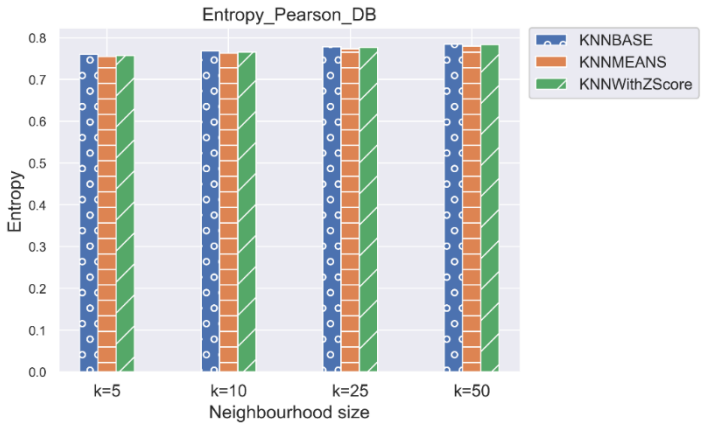
Regarding Entropy, it seems that the optimal way of dealing with popularity bias is to reduce neighbourhood sizes regardless of similarity metrics in a dense dataset such as MLM.



(a)



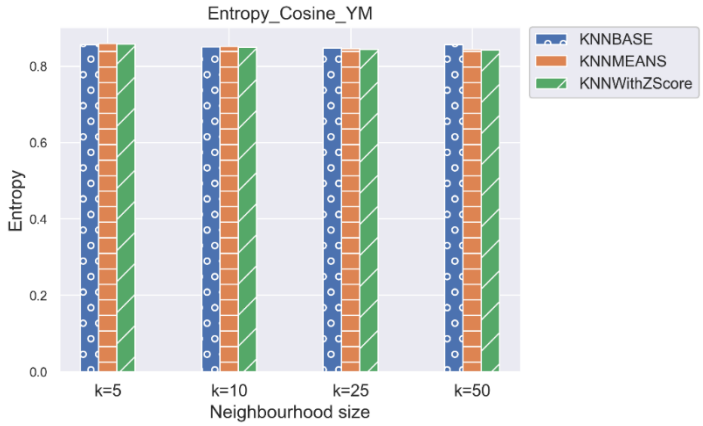
(b)



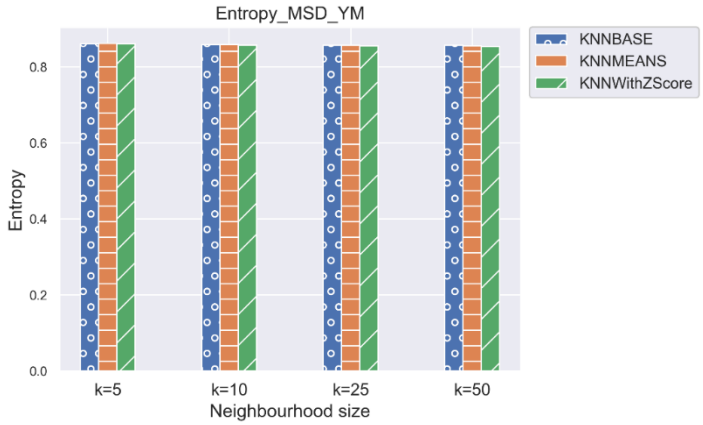
(c)

Figure 4.8 Entropy Results for Douban Books

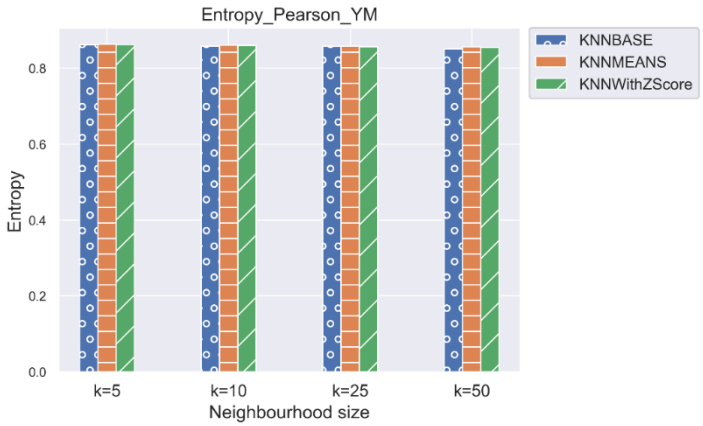
In the DB dataset, it is seen that the Cosine and MSD similarities show a negative correlation between neighbourhood sizes and Entropy values, while Pearson shows a positive correlation (Figure 4.8); it should also be noted that Cosine and MSD similarities have close Entropy values while Pearson similarity shows significantly higher Entropy values. Also, similarity metrics play roles between different recommendation algorithms rankings in terms of Entropy values. However, all different recommendation algorithms behave close to each other.



(a)



(b)



(c)

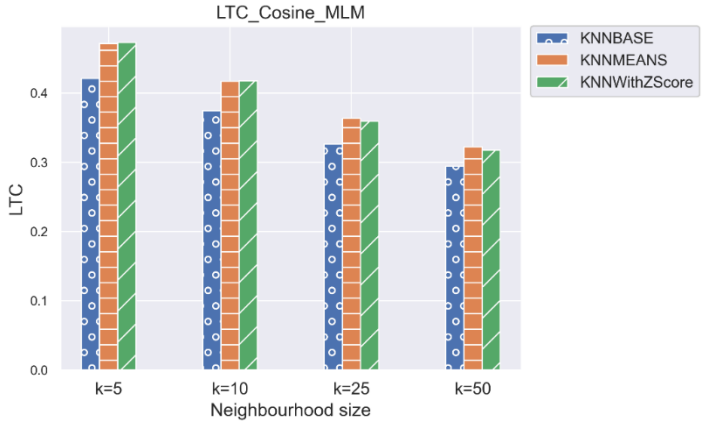
Figure 4.9 Entropy Results for Yahoo!Music

In Figure 4.9 Entropy values do not vary compared to different similarities. However, in this case, the neighbourhood size does not effect entropy levels in each similarity measure.

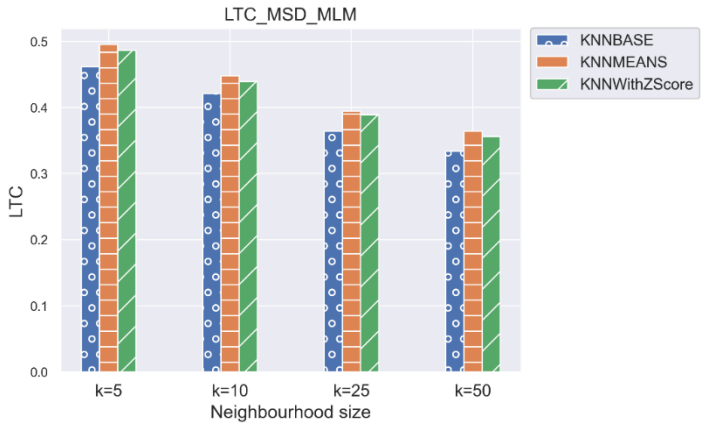
In the most sparse dataset that we have tested (YM), it can be seen that the Entropy does not get effected by our different test scenarios.

Entropy datasets need to be dense to see the effects of test cases, which shows that dealing with popularity bias might become harder in terms of using collaborative filtering with sparse datasets.

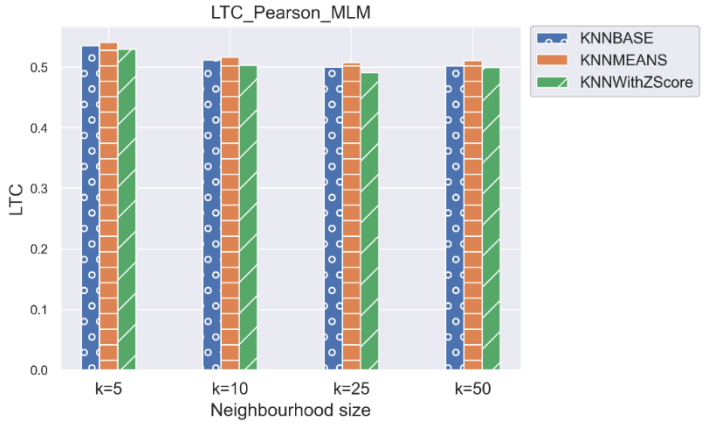
4.4 LTC Results



(a)



(b)

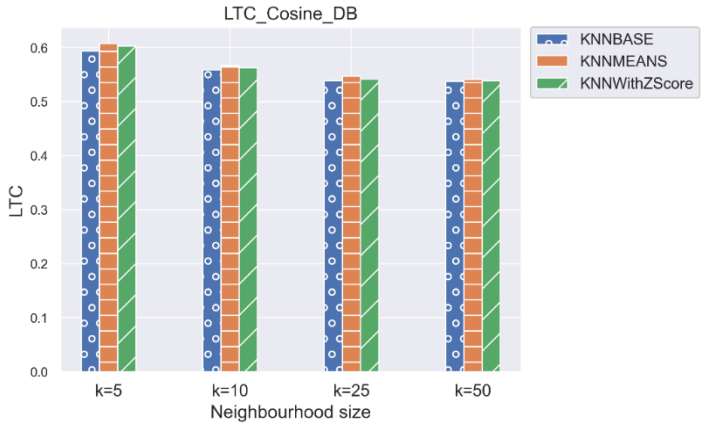


(c)

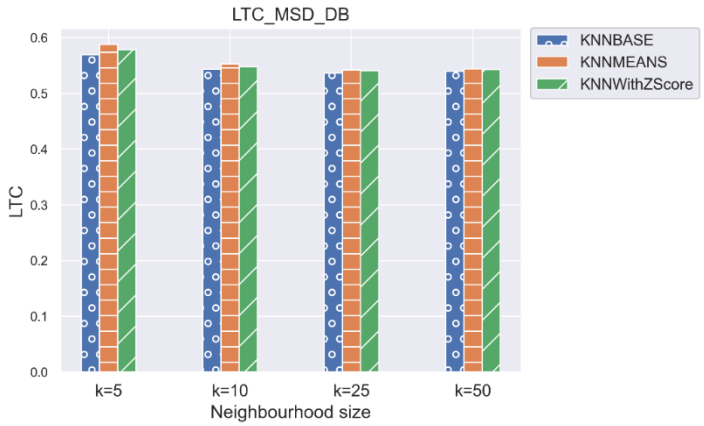
Figure 4.10 LTC Results for MovieLens-1M

In Figure 4.10 the MLM dataset shows a negative correlation between neighbourhood sizes and LTC levels. However, it can be observed that in the cosine and MSD similarities, KNNBaseline algorithms achieve significantly low LTC values compared to Pearson similarity. Also, when the LTC values are compared between the similarities, it can be seen that cosine similarity metrics LTC values are significantly lower than MSD and Pearson.

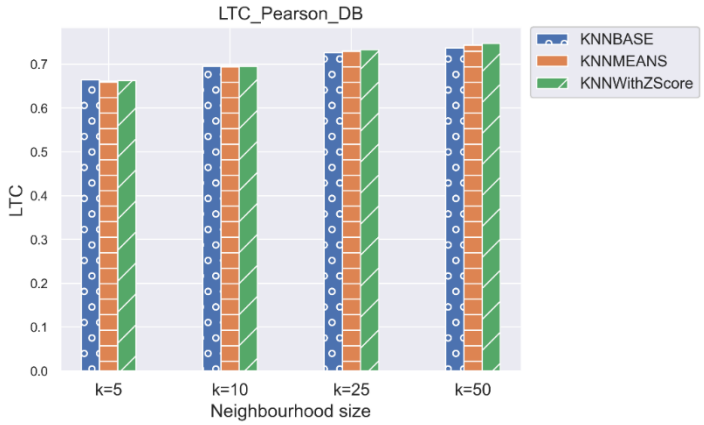
Covering the Long tail sections of the catalogue in top-N recommendation seems to be in a correlation with both similarity metrics and neighbourhood sizes; it is seen that using MSD similarity and low numbers of neighbours, almost 50% of items in top-N can become long-tail items.



(a)



(b)

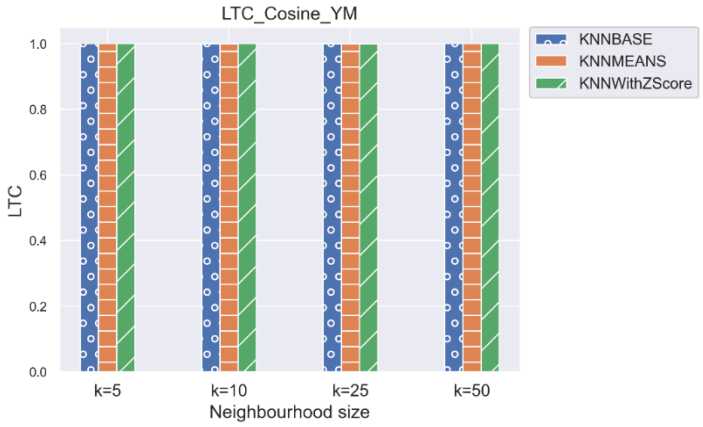


(c)

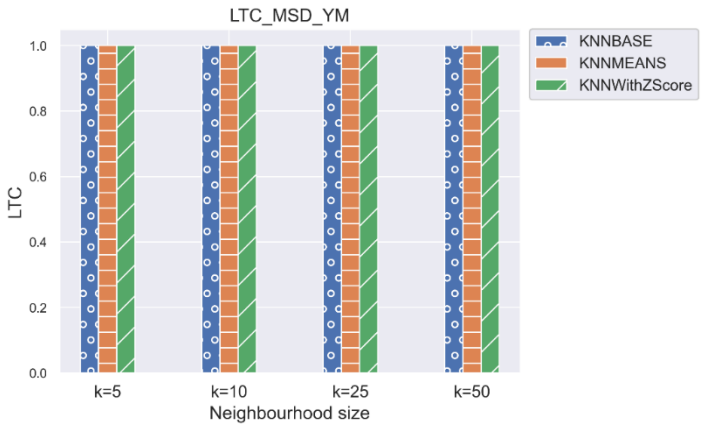
Figure 4.11 LTC Results for Douban Books

Figure 4.11 In the DB dataset, it can be observed that cosine and MSD have a negative correlation between LTC values and neighbourhood sizes, while Pearson has a positive correlation. Furthermore, it should also be noted that Pearson similarity LTC values are significantly higher than other similarities.

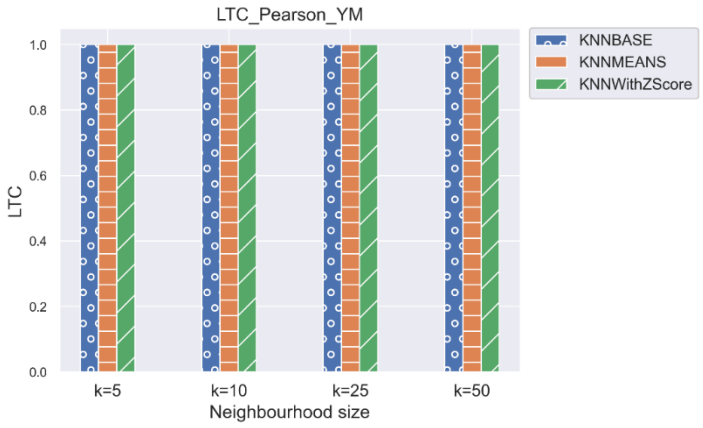
It is interesting to see that values stop changing in terms of LTC after neighbourhood sizes become more than 5, with MSD and Cosine similarities. However, with Pearson, there is quite the opposite effect; with increasing neighbourhood sizes, LTC increases significantly, which also proves that the approach for dealing with popularity bias should also be changed in each different dataset.



(a)



(b)



(c)

Figure 4.12 LTC Results for Yahoo!Music

In Figure 4.12 There can not be any correlation conserved between any of the experimental parameters. LTC value always stays 1.0 no matter what the settings are.

In the most sparse dataset that we have tested (YM), it can be seen that the LTC value does not get effected by our different test scenarios. Moreover, it always stays at 100%.

Again in LTC, it is seen that each dataset requires different approaches in terms of dealing with popularity bias.

5. CONCLUSIONS

In this study, we aimed to test the popularity bias in recommendation systems using different test scenarios with different neighbourhood sizes in collaborative filtering algorithms (recommendation algorithms), datasets, and similarity metrics. The specific algorithms that were tested were KNNBaseline, KNNWithMeans, and KNNwithZscore, which are widely used and well-established algorithms in the field of collaborative filtering. These algorithms were selected in order to provide a comprehensive understanding of the effects of popularity bias across different algorithm types. The datasets used were DB, YM, and MLM, which were chosen to represent a diverse range of characteristics, such as sparsity and density, to better understand the impact of dataset characteristics on popularity bias. The similarity metrics that were used were cosine, MSD, and Pearson, which are widely used in collaborative filtering and have been shown to have a significant impact on the performance of recommendation systems. Using these different test scenarios, algorithms, datasets and similarity metrics allowed us to understand the popularity bias in recommendation systems and how to mitigate it.

In this test, we have noticed that dealing with popularity bias is possible by using correct recommendation setups in the algorithms depending on the datasets. This means that by carefully selecting and adjusting parameters such as the neighbourhood size, similarity metric, and algorithm, it is possible to mitigate the effects of popularity bias without the need for creating a new recommendation algorithm. This is an important consideration for already running recommendation systems that are integrated into daily running platforms, as creating a new algorithm can be a costly and time-consuming process. Additionally, by understanding the characteristics of the dataset and choosing the appropriate recommendation setup, it is possible to achieve a balance between diversity and personalisation in the recommendations, which can lead to a more effective and fair recommendation system. Our research has shown that it is possible to improve the performance of recommendation systems by addressing popularity bias without the need for costly and time-consuming developments.

However, the character of the dataset itself plays an important role in recommendations popularity bias. Our study showed that in sparse datasets such as YM, no matter the setup, there was always a bias that was never affected regardless of the experiment setup. In contrast, each metric plays an important role in dealing with the popularity bias in a dense dataset such as MLM, where it shows that extra attention is required in the recommendations systems' setup of a dense dataset.

Algorithms between each other did not play much significant role in popularity biases though similarity metrics, especially the MSD similarity, in some cases changed the behaviour of the algorithm significantly.

Finally, it can be noted that using KNNBaseline in a dense dataset, with a similarity metric besides MSD similarity, can achieve a low popularity biased recommendation.

6. REFERENCES

Burke, R., Felfernig, A., & Göker, M. H. (2011). Recommender Systems: An Overview. AI Magazine, 32(3), 13-18. https://doi.org/10.1609/aimag.v32i3.2361

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In Recommender Systems Handbook (pp. 1-35). Springer.

Anderson, E. W. (1998). Customer Satisfaction and Word of Mouth. Journal of Service Research, 1(1), 5-17.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In Recommender Systems Handbook (pp. 1-35). Springer.

Schafer, J. B., Konstan, J., & Riedl, J. (1999). Recommender Systems in E-commerce. In Proceedings of the 1st ACM Conference on Electronic Commerce (pp. 158-166).

Nguyen, T. T., Harper, F. M., Terveen, L., & Konstan, J. A. (2018). User Personality and User Satisfaction with Recommender Systems. Information Systems Frontiers, 20(6), 1173-1189.

Bennett, J., Lanning, S., et al. (2007). The Netflix Prize. In Proceedings of KDD Cup and Workshop, volume 2007, page 35. Citeseer.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems (TIIS), 5(4), 1-19.

Dror, G., Koenigstein, N., Koren, Y., & Weimer, M. (2011). The Yahoo! Music dataset and KDD-Cup'11. In Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18, pages 3-18.

Paparrizos, I., Cambazoglu, B. B., & Gionis, A. (2011). Machine learned job recommendation. In Proceedings of the Fifth ACM Conference on Recommender Systems, pages 325-328.

Jawaheer, G., Weller, P., & Kostkova, P. (2014). Modeling user preferences in recommender systems: A classification framework for explicit and implicit user feedback. ACM Transactions on Interactive Intelligent Systems (TiiS), 4(2), 1-26.

Adomavicius, G., & Kwon, Y. (2011). Improving aggregate recommendation diversity using ranking-based techniques. IEEE Transactions on Knowledge and Data Engineering, 24(5), 896-911.

Burke, R., & Ramezani, M. (2011). Matching recommendation technologies and domains. In Recommender Systems Handbook (pp. 367-386).

Garcin, F., Dimitrakakis, C., & Faltings, B. (2013). Personalised news recommendation with context trees. In Proceedings of the 7th ACM Conference on Recommender Systems, pages 105-112.

Liu, J., Dolan, P., & Pedersen, E. R. (2010). Personalised news recommendation based on click behavior. In Proceedings of the 15th international conference on Intelligent user interfaces, pages 31-40.

Steck, H. (2013). Evaluation of recommendations: rating-prediction and ranking. In

Proceedings of the 7th ACM Conference on Recommender Systems, pages 213-220.

Burke, R. (2007). Hybrid web recommender systems. In The Adaptive Web (pp. 377-408). Springer-Verlag.

Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In Recommender Systems Handbook (pp. 73-105). Springer.

Pazzani, M., & Billsus, D. (2007). Content-based recommendation systems. The Adaptive Web, 325-341.

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. Communications of the ACM, 35(12), 70.

Ghazanfar, M. A., & Prugel-Bennett, A. (2010, January). A scalable, accurate hybrid recommender system. In 2010 Third International Conference on Knowledge Discovery and Data Mining (pp. 94-98). IEEE.

Huang, S. L. (2011). Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods. Electronic Commerce Research and Applications, 10(4), 398-407.

Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of library and information systems, 69(Supplement 32), 175-186.

Adomavicius, G., & Tuzhilin, A. (2005). Toward 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), 734-749.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, 12(4), 331-370.

Shani, G., & Gunawardana, A. (2011). Evaluating recommendation systems. In Recommender Systems Handbook (pp. 257-297). Springer.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In Proceedings of the fourth ACM conference on Recommender systems (pp. 39-46).

Adomavicius, G., & Tuzhilin, A. (2005). Toward 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), 734-749.

Cheng, H., Kammar, O., & Gudivada, V. N. (2016). A survey of collaborative filtering techniques. ACM Computing Surveys, 48(4), 1-36.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorisation techniques for recommender systems. Computer, 42(8), 30-37.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. Knowledge-Based Systems, 46, 109-132.

Cantador, I., Bobadilla, J., Ortega, F., & Hernández-Leal, P. (2011). A comparative study of item-based top-n recommendation algorithms. Knowledge-Based Systems, 24(4), 540-549.

Dacrema, M., Cremonesi, P., Jannach, D., & Lrecse, L. (2019). Are we really making much progress? A worried review of recent neural recommendation approaches. Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, 13-20.

Liang, Y., Li, Y., & Chen, Y. (2016). A survey of collaborative filtering techniques. ACM Computing Surveys, 48(4), 1-36.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, 285-295.

Pilászy, I., Németh, B., & Tikk, D. (2010). The performance of different similarity measures in collaborative filtering. In 2010 IEEE 10th International Conference on Data Mining (pp. 1165-1170). IEEE.

Shi, Y., Gao, J., & Kong, X. (2014). A collaborative filtering recommendation algorithm based on user similarity. Journal of Computational Information Systems, 10(5), 2363-2368.

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

Harper, F. M., & Konstan, J. A. (2016). The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS), 5(4).

Steck, H., Caraciolo, M., & Moreira de Oliveira, J. P. (1995). MovieLens: A Movie Recommendation System. In IEEE International Conference on Multimedia Computing and Systems (ICMCS).

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

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Terveen, L. G., & Riedl, J. T. (1997). GroupLens: An open architecture for collaborative filtering of netnews. In Proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW).

Douban (2021). Douban website. Retrieved from https://www.douban.com/

Li, Y., Liu, Y., Zhang, M., & Chen, E. (2010). Personalised recommendation using social information on the douban.com website. In 2010 IEEE International Conference on Data Mining (pp. 977-982). IEEE.

Chen, E., & Chen, X. (2012). A social recommendation model based on user-generated contents. Expert Systems with Applications, 39(2), 1393-1401.

Zhou, T., Zhu, X., & Song, Y. (2014). A collaborative filtering recommendation algorithm based on social information. Expert Systems with Applications, 41(11), 5423-5430.

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009). BPR: Bayesian personalised ranking from implicit feedback. In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence (pp. 452-461).

Lu, L., & Karypis, G. (2012). Evaluation of recommendation systems. In Recommender systems handbook (pp. 257-297). Springer, Boston, MA.

Zhang, Y., Liu, J., & Chen, L. (2015). A long-tail recommendation approach based on semantic similarity. Expert Systems with Applications, 42(23), 9474-9485.

Li, Y., Li, X., & Yin, J. (2016). A hybrid long-tail recommendation algorithm based on collaborative filtering. Expert Systems with Applications, 55, 64-71.

Zhang, Y., & Lattner, C. (2018). Is Your Model Biased? An Empirical Study of Unbiased Learning. In International Conference on Machine Learning and Data Mining in Pattern Recognition (pp. 391-404). Springer, Cham.

Saha, A., & Wu, X. (2018). Long-Tail Coverage: A New Metric for Evaluating Recommender Systems. In Proceedings of the 2018 World Wide Web Conference (pp. 1589-1598). International World Wide Web Conferences Steering Committee.

Shannon, C. E. (1948). A mathematical theory of communication. Bell System Technical Journal, 27(3), 379-423.

Lu, L., & Karypis, G. (2012). Evaluation of recommendation systems. In Recommender systems handbook (pp. 257-297). Springer, Boston, MA.

Sanders, R. (1987). The pareto principle: its use and abuse. Journal of Services Marketing, Vol. 1. No. 1, pp. 30-35. Jannach, D., Ludewig, M., & Jäschke, R. (2011). Beyond accuracy: evaluating recommender systems by coverage and serendipity. In Proceedings of the fifth ACM conference on Recommender systems (pp. 257-260).

Zhang, Y., Lathia, N., & Liu, H. (2014). Diversity in recommendations: the long tail of the users' preferences. In Proceedings of the seventh ACM conference on Recommender systems (pp. 125-132).

Koren, Y., & Bell, R. (2015). Cold-start recommendations in recommender systems. In Advances in Recommender Systems (pp. 305-322). Springer, Cham.

Gullo, F., Xiang, S., Kazai, G., & Jäschke, R. (2016). Recommender systems with diversity-promoting similarity measures. In Proceedings of the 39th international ACM SIGIR conference on Research and Development in Information Retrieval (pp. 671-680).

Gao, H., Yang, J., & Chen, Y. (2018). Addressing cold-start in recommendation with trust. In Proceedings of the 2018 World Wide Web Conference (pp. 1389-1398). International World Wide Web Conferences Steering Committee.

Yoshida, K., Hara, K., & Takahashi, S. (2020). Addressing popularity bias in recommendation: a two-fold approach. In Proceedings of the 25th International Conference on Intelligent User Interfaces (pp. 322-327).

Bilge, E., & Güneş, Ş. (2019). Diversity in personalized search. In Proceedings of the 42nd international ACM SIGIR conference on Research and Development in Information Retrieval (pp. 811-820).

Bilge, E., & Güneş, Ş. (2020). Examining the impact of diversity in recommendation on user satisfaction. In Proceedings of the 2020 CHI Conference on Human Factors

in Computing Systems (pp. 1-15).

Bilge, E., & Güneş, Ş. (2020). Combating popularity bias in personalized search. In Proceedings of the 43rd international ACM SIGIR conference on Research and Development in Information Retrieval (pp. 991-1000).

ÖZGEÇMİŞ

OSMAN ALPER MISIRLI

ÖĞRENİM BİLGİLERİ

Yüksek Lisans 2020-2023	Akdeniz Üniversitesi Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği Anabilim Dalı, Antalya
Yüksek Lisans 2019-2020	Orta Doğu Teknik Üniversitesi Enformatik Enstitüsü, Çoklu ortam Bilişimi (Oyun Teknolojileri), Ankara
Lisans (Erasmus+) 2016-2017	Politechnika Krakowska im. Tadeusza Kościuszki Mühendislik Fakültesi, Bilgisayar Mühendisliği, Krakow
Lisans 2015-2019	Anadolu Üniversitesi Mühendislik Fakültesi, Bilgisayar Mühendisliği, Eskişehir

MESLEKİ GÖREVLER

Bilgisayar Mühendisi 2021-Devam Ediyor	AYASİS YAZILIM VE BİLİŞİM TEKNOLOJİLERİ A.Ş. MentalUP ,Unity Developer,Yıldız Teknik Üniveristesi Teknopark Davutpaşa İstanbul