Minimizing the Long-tail Problem in Collaborative Filtering Based Recommender Systems Using Clustering

Yash Mundhra
Supervisor(s): Aleksander Czechowski, Frans Oliehoek, Oussama Azizi, Davide Mambelli
EEMCS, Delft University of Technology, The Netherlands

24-6-2022

Abstract

Recommender systems are an essential part of online businesses in today's day and age. They provide users with meaningful recommendations for items and products. A frequently occurring problem in recommender systems is known as the long-tail problem. It refers to a situation in which a majority of the items in the data set have limited ratings due to which many recommender systems, especially collaborative filtering based methods, are not able to recommend these items, also known as long-tail items. Although popular items are easier to recommend, it has been noticed that long-tail items often generate a significant fraction of the revenue and therefore should also be recommended to users. This paper proposes a modified version of a collaborative filtering based recommender system aimed to reduce the effects of the long-tail recommendation problem (LTRP). The algorithm first splits the data set into the head H and the tail T and clusters the items from the tail. The average rating avg for each cluster is calculated and for all users and their unrated long-tail items, the rating for that item is set to avg with a probability of p. Now the standard collaborative filtering algorithm is run with the newly inserted ratings. The inserted ratings reduce the sparsity of the data set and therefore make it easier to recommend long-tail items. Empirical experiments on the 100K MovieLens data set indicate that the proposed algorithm recommends more long-tail items than the standard collaborative filtering algorithm, thus reducing the effects of the LTRP while maintaining the same or a slightly lower accuracy of the recommender system.

1 Introduction

With the rapid growth of technology and the increased use of e-commerce platforms the need to recommend relevant items to users has never been as important. Recommender systems cater to this exact need of user recommendations. The purpose of a recommender system is to generate attractive recommendations for items and products to users. Movie recommendations on Netflix, personalized playlist recommendations on Spotify, and product recommendations on Amazon are examples of recommender systems being used in our daily lives. Although the definition of recommender systems has evolved since it was first proposed by Goldberg et al. in 1992 [10] it can be broadly defined as "a system that has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" [9].

Over the years, several different algorithms for recommender systems have been developed. The content-based and collaborative filtering (CF) based recommendation methods are the two traditional methods used for producing recommendations. In the past few years, several machine learning and deep learning based adaptations have also been made to these classical methods. An underlying problem that is faced by a majority of the algorithms is known as the 'long-tail problem'. The long-tail problem refers to a situation in which "recommender systems ignore unpopular or newly introduced items having only a few ratings and focus only on those items having enough ratings to be of real use in the recommendation algorithms" [17].

Items that are found in the long-tail of the data set are known as long-tail items while all other popular items that are not in the long-tail are known as the short-head items. Long-tail recommendations can increase the diversity, coverage, and serendipity of a set of recommendations while still being relevant or even more relevant for users than short-head items. Additionally, for businesses, long-tail items often have large marginal profits in comparison to short-head items, which means that long-tail recommendations can also be more profitable for companies [13].

Several studies have been conducted on how to reduce the effect of the long-tail problem in the context of recommender systems. Various graph based methods were proposed that represent interactive data between users and items in the form of a graph. Yin et al. [20] discussed a method involving a 'bipartite graph'. Johnson and Ng [12] proposed an extension of the 'bipartite graph method' known as the 'tripartite graph method'. Luke et al. [14] combined the two methods and created an 'extended tripartite graph method'. Various research papers have also considered the use of deep learning for long-tail item recommendations. Bai et al. [8] proposed a deep learning framework for long-tail item recommendations known as the DLTSR. Sreepada and Patra [18] used few-shot learning techniques to solve the long-tail problem.

The use of clustering to alleviate the long-tail of recommender systems has also been explored in several research papers. Park and Tuzhilin [17] proposed the EI (Each Item), TC (Total Clustering), and CT (Clustered Tail) long-tail recommendation methods in the context of machine learning based recommender systems. Park [16] also introduced the idea of AC (Adaptive Clustering) in which the degree of clustering is based on how often an item is rated.

The long-tail problem is a problem that occurs mainly in collaborative filtering based recommender systems [20]. The paper by Park and Tuzhilin mentions that a possible extension to their research would be to incorporate the CT (Clustered Tail) method in a collaborative filtering based recommendation system. This research will therefore investigate whether the CT method can be applied to collaborative filtering based recommender systems and what influence this has on the performance of the recommender system. Additionally, the research will investigate how the number of clusters and the cutting point impacts the performance of the recommender system. As a consequence, the research question can be defined as follows *'To what extent can clustering be applied to the long-tail of collaborative filtering recommender systems such that more long-tail items are included in the set of recommendations while not affecting the accuracy of the recommender system and how do the number of clusters and the cutting point have an influence on this?'*

The remainder of the paper is organized as follows. Section 2 provides the background of the collaborative filtering based recommender system and the long-tail recommendation problem. Section 3 takes a more detailed look at the solution that is being proposed and describes how the experiments are conducted. Next, section 4 describes the experimental setup and the results gathered through the experiments. The analysis of the results, a discussion about the difficulties and future extensions of the work are presented in section 5. A summary of the study is given in section 6. Finally, section 7 discusses the responsible research practices that were followed during this study.

2 Background

The following section dives deeper into the topics covered in this study. Section 2.1 describes the working behind a collaborative filtering based recommender system. Section 2.2 dives deeper into the long-tail recommendation problem that is being tackled in this research paper.

2.1 Collaborative Filtering Based Recommender System

The recommendation problem can be formulated as a maximization problem that maximizes the user's utility u. Let u be a function that measures the usefulness of an item s to a user c, i.e $u : C \times S \Rightarrow R$. Within this formal notation, C is the set of all users, S is the set of all possible items that can be recommended and R is a totally ordered set such as non-negative numbers or real numbers within a range. The goal of a recommender system is to choose an item $s' \in S$ for each user $c' \in C$ such that the users utility is maximized [2].

Collaborative filtering is one of the most prominent and popular algorithm used for recommender systems. The main idea behind collaborative filtering is that users who have had similar preferences in the past will continue to behave similarly in the future. Therefore the algorithm tries to find users who have similar preferences or ideas, and make recommendations based on this knowledge. The algorithm can be categorized into memory-based CF and model-based CF. The model-based algorithm learns a predictive model from the rating data to make predictions about rankings of items to each user. The memory-based algorithm generates recommendations using the rating data directly [19].

The memory-based collaborative filtering algorithm starts off with finding the pairwise similarities between all users. The similarity between users is measured in terms of the ratings given by them on the movies. Several different similarity metrics can be used to find the similarity between two users such as the Pearson correlation coefficient, Jaccard similarity, Euclidean distance, and Manhattan distance. The similarity metric used in this study is **cosine similarity** since it is the most used metric for collaborative filtering based recommender systems [3]. Mathematically, the cosine similarity calculates the cosine of the angle between two vectors in a multi-dimensional space. It is defined as follows:

$$r = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \, ||\vec{y}||} \tag{1}$$

where:
$\vec{x}$ = rating vector for user x
$\vec{y}$ = rating vector for user y

Once the pairwise similarity between all users has been calculated, the unknown ratings of an item by a user can be predicted using a weighted average of the ratings given by similar neighbors. The formula for predicting the ratings [4] is defined as follows:

$$R_U = \frac{\sum_{i=1}^{n} R_u * S_u}{\sum_{i=1}^{n} S_u} \tag{2}$$

where:
R_U = Predicted rating by user U
n = Number of nearest neighbour
R_u = Rating of item by neighbour u
S_u = Similarity factor between neighbour u and user U

The number of neighbors n that is included in the calculation depends on how the neighborhood set is selected. Neighborhood selection can have a significant impact on the performance of the collaborative filtering recommender system. There are several different approaches that can be followed to determine which neighbors to include when calculating the predicted rating. If the data set is small enough it can be chosen to include all users in the set of neighbors. This increases the computational time of the algorithm for larger data sets and was therefore not chosen. With the *threshold* approach, a user is added to the neighborhood set when the similarity score between the two users is greater than a predefined threshold T. A disadvantage of this method is that some predictions may have a larger neighborhood set than others making some predictions more accurate than others. The approach used in this study is known as *Top-N* where the n most similar users are added to the set of neighbors. In a paper by Bahadorpour et al. [7] the optimal number of neighbors on which the recommender system would perform well on the MovieLens data set was between 10 and 15. Therefore the value of n chosen for the conducted experiments is 10.

2.2 Long-Tail Recommendation Problem

The long-tail is a concept that has been studied by statisticians and scientists for a long time and refers to a set of products not commonly used by consumers. By Pareto's principle, it can be said that 80% of the sale comes from only 20% of the stock. The term "long-tail" was popularized by Chris Anderson through a magazine article in 2004 [6]. In his book "The Long Tail: Why the Future of Business Is Selling Less of More" he discusses how the rise in technological advancement and unlimited shelf space has made it easier for consumers to buy niche products instead of popular products which resulted in the shift from the *hit market* into the *niche market*. Figure 1 shows the transformation from the hit market to the niche market.

Anderson theorized that products in low demand (long-tail items) can collectively make up a market share that rivals or exceeds the few best-selling items. To create a successful long-tail business, however, a recommender system must be able to recommend long-tail items to users. Unfortunately, traditional recommender systems tend to recommend only

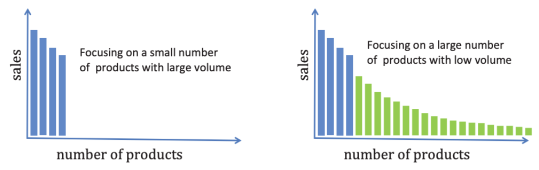

Figure 1: Hit vs Niche Markets [20]

popular items, this is known as the long tail recommendation problem (LTRP).

To reduce the effects of the LTRP, Park and Tuzhilin [17] proposed two new recommendation methods for machine learning based recommender systems. The total clustering (TC) method "clusters the whole item-set S into different sets and builds rating predictive models for each resulting group". The clustered tail (CT) method splits the data into the head H and tail T. Clustering is applied to the tail T while the items in the head are left un-clustered. Predictive models are built for each cluster in the tail and the individual items in the head. A few years later Park introduced a new way of clustering items to reduce the effects of the LTRP known as adaptive clustering (AC). The AC method clusters the items based on their rating frequency. If an item has a low rating frequency it is clustered more whereas items with a higher cluster frequency are clustered less [16].

There are several different clustering algorithms that can be used for clustering the items in the tail together. The method used in this study is the centroid based k-means algorithm. The algorithm selects k different centroids and groups items based on their proximity to the centroids such that the sum of squares within a cluster is minimized [5].

3 Methodology

This study investigates how a variation of the clustered tail (CT) method proposed by Park and Tuzhilin [17] can be applied to collaborative filtering based recommender systems and whether this could improve the performance of recommender systems and reduce the effect of the LTRP. The modified version of the clustered tail recommender system will be referred to as the clustered tail collaborative filtering recommender system (CTCF-RS)

The algorithm starts by splitting the set of items into two sets the head H and the tail T. A value for the cutting-point c is chosen which dictates in which one of the two sets an item gets placed. Items with a rating frequency (number of times it has been rated) of more than c belong to the head H and items with a rating frequency less than c belong to the tail T. The rating frequency of an item refers to the number of times an item has been rated. Items from the tail T are clustered using a clustering method. Items from the head H do not get clustered at all.

Given that the items from the tail T have been clustered, the average rating avg for all items in each cluster is computed. For each user in the data set and all of their unrated long tail, the rating for that item is set to the avg rating of the cluster it belongs to with a probability of p. With a probability of $1-p$, the movie remains unrated.

Once all users have been considered and the new ratings have been inserted the standard collaborative filtering algorithm is executed on the data set with the newly inserted ratings. The insertion of new ratings reduces the sparsity of the data set allowing the CF algorithm to be more balanced. Instead of only focusing on the popular items the algorithm now considers long-tail items as well.

4 Experiment

In order to validate the proposed solution, several experiments were conducted. The following section details the experimental settings and presents the results of the experiments. Section 4.1 includes an overview of the data used, performance measurements, statistical tests, and the variables from the experiment. Section 4.2 visualizes and describes the results that were found from the conducted experiments.

4.1 Experimental Setup

Data Set

The data set that was chosen for the experiments is the MovieLens 100K dataset [1] [11]. The data set consists of 100,000 ratings from 1000 users on 1700 movies collected in a period from September 19th, 1997 through April 22nd, 1998. For each movie, the title, release date, IMDb URL, and genre are given. For each user, the age, gender, occupation, and zip code is given. Every user in the data set has rated at least 20 movies. The ratings are given on a scale from 1 to 5. The data set has a density of 6.30% meaning that a majority of users have not rated most movies. Figure 2 illustrates the distribution of the number of ratings received for all the movies, a clear long tail distribution can be seen making it an ideal choice to perform experiments on.

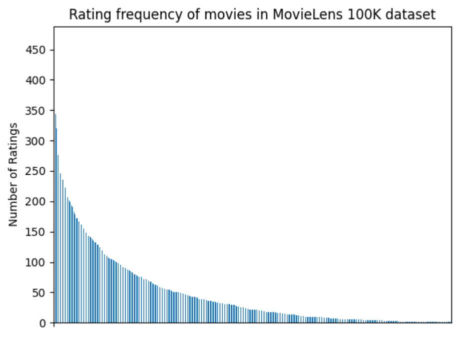

Figure 2: Rating frequency (number of times a movie has been rated) of movies in the MovieLens 100K data set.

Performance Metrics

The effect of the proposed solution and the influence of the hyper parameters on the CTCF recommender system was measured using four different performance metrics. The performance was measured using the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), Diversity, and

[1]https://grouplens.org/datasets/movielens/100k/

Coverage. The RMSE and MAE measure the accuracy of the recommender system to predict unknown ratings whereas the diversity and coverage are metrics specifically geared towards the long-tail problem.

RMSE: The root mean squared error measures the error of a model in predicting quantitative data. It can be formally defined as follows

$$RMSE = \sqrt{(\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2} \tag{3}$$

where:
n = total number of items that need to be predicted
y_i = actual rating for the item
x_i = predicted rating for the item

MAE: The mean absolute error measures the average magnitude between the measured value and the 'true value'. It can be formally defined as follows

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{4}$$

where:
n = total number of items that need to be predicted
y_i = actual rating for the item
x_i = predicted rating for the item

Diversity: Diversity is a measure of how different the recommended items are from each other. A higher diversity is often beneficial for a recommender system as long as it does not impact the accuracy of the predictions. A recommender system that recommends more long-tail items does not always have a higher diversity since the items in the long tail could be similar to the ones in the short head. There are several ways for measuring diversity. The method that is followed in this research is as follows

$$diversity = 1 - \frac{\sum_{u=1}^{n} sim_{25}}{n} \tag{5}$$

where:
sim_{25} = similarity between the top 25 recommended items
n = total number of users

The similarity between the top 25 recommended items sim_{25} is calculated using the cosine similarity with the formula given in section 2.1

Coverage: Coverage refers to the percentage of items recommended by the recommender system from all possible items. The goal is to maximize the coverage since a higher coverage indicates that more items from the long-tail of the data set are predicted. The formula is given as follows

$$coverage = \frac{i_s}{i} \tag{6}$$

where:
i_s = number of distinct items recommended by the system from the top 25 recommended items per user
i = total number of items in the inventory

Clustering Items in the Tail
The movies from the tail T were clustered using an unsupervised machine learning algorithm known as K-Means clustering. The movies were clustered based on the genre as provided in the data set. It was considered to use additional variables such as release year, actor, and director to cluster the movies however, this would require the parameters to be encoded which would expand the dimensionality of the data set significantly. Six different values for the number of clusters were chosen and experimented with; 5, 10, 15, 20, 25, and 30.

Cutting Point c
The cutting point in the clustered tail algorithm determines whether an item belongs to the head H or the tail T. If the rating frequency of an item is greater than or equal to the cutting point value then it is classified as a 'head item', otherwise it is classified as a 'tail item'. The value of the cutting point could influence the performance of the recommender system as it dictates which items are clustered using the algorithm and which items are left out. Five different values for the cutting point have been chosen; 30, 50, 70, 90, and 110.

Probability for Inserting a Rating p
As mentioned in the methodology (section 3), for each user and all their unrated long-tail items, with a probability of p the rating for that item is set to the average rating avg of the cluster to which that item belongs to. With a probability of $1 - p$, the item is left unrated. The chosen value for p is 0.15. A low value is chosen to ensure that the data set does not get too populated with these ratings, as they are not an accurate representation of the ratings of the user. However, they do help with reducing the sparsity within the data set.

4.2 Experimental Results

To verify the robustness, generalizability, and accuracy of the gathered data, a 5-fold cross-validation was performed in the experiments. Cross-validation divides the data into two segments; a training set and a validation set. For each fold of the 5-fold cross-validation, a disjoint validation set was chosen. The final data was gathered by averaging the results from all five experiments. Since data was gathered for five different cutting points and six cluster values the total number of experiments grew to 150 for each performance metric.

Table 1 below shows the lowest root mean squared error that was observed for each cutting point and the baseline recommender system.

CuttingPoint_NumberOfClusters					
BaseLine	*30_15*	*50_25*	*70_20*	*90_25*	*110_30*
1,0333	**1,0187**	1,0207	1,0236	1,0252	1,0287

Table 1: Minimum RMSE for all five cutting points in CTCF-RS and baseline recommender system

As seen from table 1, the RMSE of the baseline collaborative filtering recommender system was higher than all iterations of the CTCF-RS suggesting that the CTCF-RS always has a better accuracy than the baseline recommender system. The lowest RMSE that was observed for the CTCF-RS was where the cutting point was 30 and the number of clusters was 15. At cutting point 110 and number of clusters 5 the RMSE was the highest amongst all other experiments. The number of clusters that yielded the lowest RMSE for the cutting points 50, 70, 90, and 110 was 25, 20, 25, and 30 respectively.

Figure 3 displays a heat map with the root mean squared errors of the CTCF-RS with varying values for the cutting point and numbers of clusters. An increasing trend in terms of the RMSE can be seen in the heat map. As the value for the cutting point increases the RMSE value also increases and the accuracy of the recommender system decreases. Overall the lowest value for the cutting point (i.e 30) performs the best regardless of the number of clusters. Appendix A visualizes the RMSE of the baseline and CTCF recommender system in a bar graph.

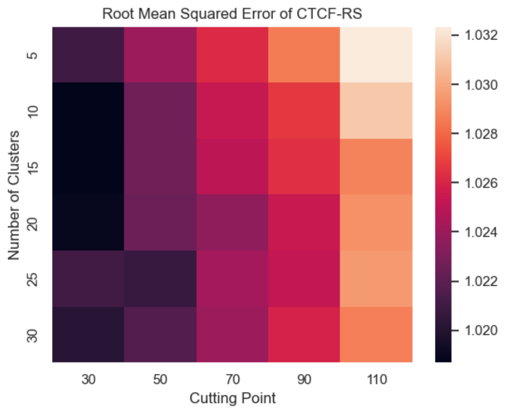

Figure 3: RMSE of CTCF recommender system with varying values for cut point and number of clusters.

Table 2 below shows the minimum mean absolute error for each cutting point and the baseline recommender system.

CuttingPoint_NumberOfClusters					
BaseLine	*30_25*	*50_25*	*70_20*	*90_25*	*110_15*
0,8161	**0,8106**	0,8136	0,8164	0,8178	0,8204

Table 2: Minimum MAE for all five cutting points in CTCF-RS and baseline recommender system

Unlike the RMSE, the MAE of the baseline recommender system is lower than the CTCF-RS for cutting points 70, 90, and 110. The MAE lies approximately in the middle of all values found. The lowest MAE that was observed from all iterations of the experiment was where the cutting point was 30 and the number of clusters was 25. This is in contrast to the findings from the RMSE which suggested that 15 is the optimal number of clusters when the cutting point was set at 30. At cutting point 110 and number of clusters 5 the MAE was the highest amongst all other experiments, the same observation was made for the RMSE. In the scenarios where the cutting point was set at 50, 70, and 90 the optimal number of clusters remained the same as was derived from the RMSE results, namely 25, 20, and 25. Finally, when the cutting point was set at 110 the number of clusters at which the MAE was minimal was found to be 15 unlike the findings from the RMSE.

Figure 4 displays the accuracy of the CTCF recommender system in terms of MAE for varying values of cutting point and number of clusters in a heat map. Similar to the RMSE an increasing trend in terms of the MAE is seen when the cutting point value increases. The heat maps of the two metrics are almost identical. Appendix B also visualizes the MAE of the baseline and CTCF recommender system together in a bar graph.

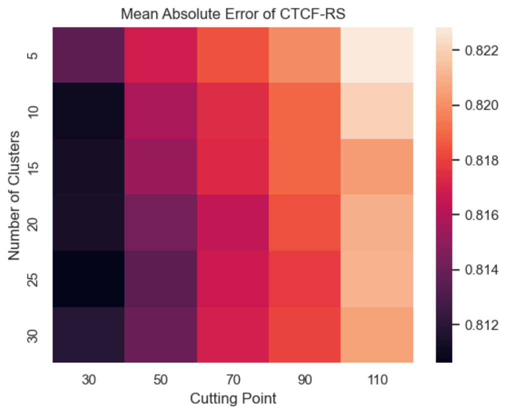

Figure 4: MAE of CTCF recommender system with varying values for cutpoint and number of clusters.

While the goal for the root mean squared error and the mean absolute error was to minimize the value, the goal for the diversity and coverage is to maximize it. Table 3 shows the maximum diversity that was found for the baseline recommender system and the CTCF-RS at each cutting point.

CuttingPoint_NumberOfClusters					
BaseLine	*30_30*	*50_30*	*70_30*	*90_25*	*110_10*
0,7279	**0,7607**	0,7347	0,7165	0,7038	0,6924

Table 3: Highest diversity for all five cutting points in CTCF-RS and baseline recommender system

It is seen from table 3 that the diversity tends to decrease as the cutting point value increases. The maximum value was observed in the scenario where both the cutting point and the number of clusters is 30. The minimum diversity was noted in the scenario where the cutting point is 110 and the number of clusters is 5. The number of clusters that gave the highest diversity for the cutting points 50, 70, 90, and 110 were 30, 30, 25, and 10 respectively. The diversity of the baseline recommender system was already quite good and therefore only the CTCF-RS with cutting point values of 30 and 50 were able to outperform the baseline recommender system in terms of diversity.

Figure 5 graphically displays the diversity of the CTCF recommender system for multiple values of the cutting point and number of clusters. As noted earlier a decreasing trend can be seen in the heat map where the diversity of the recommender system drops as the cutting point value increases. Appendix

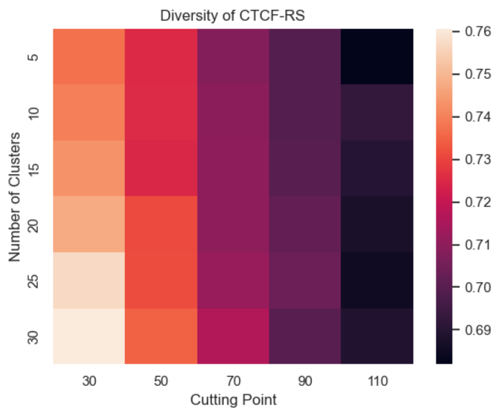

Figure 5: Diversity of CTCF recommender system with varying values for cutpoint and number of clusters

C also includes a bar chart visualizing the collected data of the baseline recommender system and the CTCF-RS.

Table 4 shows the maximum coverage value that was found for the baseline recommender system and the CTCF-RS at each cutting point value.

CuttingPoint_NumberOfClusters					
BaseLine	*30_15*	*50_15*	*70_20*	*90_10*	*110_15*
0,4013	0,4328	0,4382	0,4637	0,4751	**0,5314**

Table 4: Highest coverage for all five cutting points in CTCF-RS and baseline recommender system

The data from table 4 indicates that there is a sharp increase in the coverage of the recommender system as the cutting point value increases. The maximum coverage that was observed was in the scenario where the cutting point is 110 and the number of clusters is 5. The coverage was lowest in the situation where both the value for the cutting point and the number of clusters is 30. The coverage of the baseline recommender system stands out immediately as it is the lowest in comparison to the CTCF-RS.

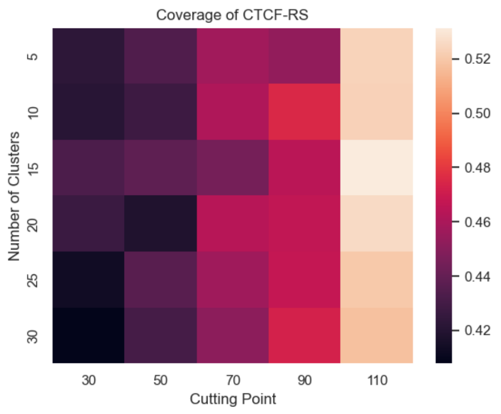

Figure 6: Coverage of CTCF recommender system with varying values for cutpoint and number of clusters

Figure 6 presents the coverage data for the CTCF recommender system in a heat map. Unlike the diversity, a positive trend can be seen in the coverage graph where the coverage increases upon an increase in the cutting point value. The increase is gradual for the first few cutting point values however the jump between the cutting point value of 90 and 110 is quite big. Appendix D visualizes the coverage data associated with the baseline recommender system and the CTCF-RS in a bar chart.

5 Discussion

The following section performs an analysis of the experimental results that were presented in section 4.2. Additionally, the difficulties that were encountered during the experimentation, limitations of these experiments, and future work that can be done on this topic will also be discussed in Section 5.1.

The accuracy of the recommender system in terms of the root mean squared error was improved by the CTCF-RS. The data suggests that there is a negative correlation between the accuracy of the recommender system and the cutting point. An increase in the cutting point value resulted in a decrease in the accuracy of the system. The accuracy was the highest when the cutting point was set at 30. This suggests that movies that have a rating frequency greater than 30 can be predicted accurately by the baseline recommender system however movies with a rating frequency lower than 30 benefit from the CTCF algorithm. As the cutting point increases the algorithm includes movies that have been rated often in the tail and the performance of the algorithm decreases.

It was also observed that the optimal number of clusters for all values of the cutting point lies between 15 and 30. This suggests that a lower number of clusters is unfavorable for the accuracy of the recommender system. This is likely because a lower number of clusters will tend to cluster dissimilar movies together making the injected rating inaccurate and thereby reducing the accuracy of the system. If the number of clusters would be too high then the number of movies in each cluster would be too low which would also be disadvantageous to the accuracy of the recommender system. Therefore it is important to find the optimal value for the number of clusters.

The mean absolute error showed a similar pattern to the RMSE of the CTCF-RS. As the value for the cutting point would increase the MAE would increase i.e the accuracy of the system would decrease. The optimal number of clusters was narrowed down even further by the MAE being between 15 and 25. In contrast to the RMSE, the MAE of the baseline recommender system was also fairly good and even better than the CTCF-RS for cutting points higher than or equal to 70. It can be hypothesized that the RMSE of the baseline recommender system was higher than the CTCF-RS unlike the MAE because the RMSE penalizes more for larger mistakes that may have occurred in the baseline predictions.

Similar to the RMSE and the MAE, the diversity graph showed a decreasing trend suggesting a negative correlation between the diversity and the cutting point. As the cutting point value increased the diversity of the recommender system decreased. The diversity of the baseline recommender system was quite good already and therefore outperformed the CTCF-RS when the cutting point was greater than or equal to 70. In the scenario where the cutting point value was 30 the diversity of the CTCF recommender system was always greater than the baseline recommender system regardless of the number of clusters. For the cutting point value of 50, the CTCF-RS outperformed the baseline recommender system occasionally depending on the number of clusters.

It is seen from the experimental data that the diversity is highest for larger values of the number of clusters. The negative correlation between the diversity and cutting point is likely caused because too many movies are placed in the long tail due to which ratings are injected for a large number of movies. This results in many of the same movies being recommended and thereby reducing the diversity of the recommended items.

The CTCF recommender system shows a significant improvement in the coverage in comparison to the baseline recommender system (approximately 13% higher coverage). For any combination of cutting points and number of clusters, the CTCF-RS outperforms the baseline recommender system. In contrast to the previous three performance metrics, the experimental results suggest a positive correlation between the coverage and the cutting point. An increase in the cutting point value results in an increase in the coverage of the recommender system. A point to be noted from the data is that there is a significant jump in the coverage between cutting points 90 and 110.

The positive correlation between the cutting point and the coverage is because an increase in the cutting point value results in more movies being part of the long tail item set. Therefore the number of movies for which ratings are injected increases and as a result, the recommender system is able to recommend more items from the long tail rather than recommending items only from the short head. This also explains the big jump in coverage between cutting points 90 and 110, a lot of movies were included in the long tail when the cutting point was shifted to 110.

The CTCF recommender system performed best in terms of the RMSE, MAE, and diversity when the value of the cutting point was set at 30. However, the coverage of the CTCF recommender system was highest at a value of 110 for the cutting point. Therefore, to accommodate for all four performance metrics a trade-off must be made. Figure 7 is a scatter plot showing the relationship between RMSE and Coverage of the CTCF-RS. The scatter plot visualizes the trade-off between the accuracy and coverage more clearly. The coverage of the recommender system increases upon a decrease in accuracy.

As observed earlier and confirmed by the scatter plot in figure 7 the improvement in coverage between cutting points 90 and 110 is quite significant. All data points with cutting point 110 are separated from the rest of the data points. To find the optimal parameters for the recommender system a data point on the Pareto front must be chosen. All points on the Pareto frontier are such that an improvement in one metric will always result in a decrease in the other metric. Therefore the points on the Pareto frontier are the most optimal points.

Overall, the CTCF-RS managed to outperform the performance of the baseline recommender system in all metrics.

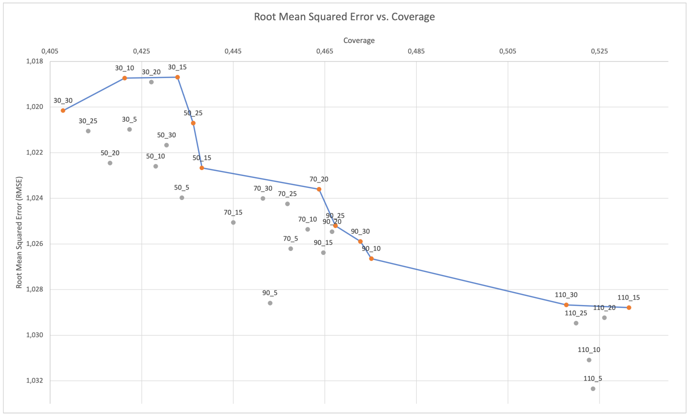

Figure 7: Scatter Plot of RMSE against Coverage of CTCF-RS

Additionally, the algorithm ensures that more items from the long tail are recommended, thereby reducing the effects of the LTRP.

5.1 Difficulties, Limitations and Future Extensions

While conducting the experiments several difficulties and limitations were encountered. Firstly it was noticed that the run-time of the CTCF-RS was significantly longer than that of the baseline recommender system. This was because the CTCF-RS iterates over all products for all users making the run time of the algorithm $\mathcal{O}(n^2)$. It is therefore an important consideration to make whether the improvement in accuracy, diversity, and coverage are worth the trade-off that is made to the run-time. Future work can explore whether there is a way to reduce the run time of this algorithm.

Although the CTCF-RS performs better than the baseline recommender system for all performance metrics, it has only been tested on a single data set making it harder to draw definitive conclusions. A future extension to this research would therefore be to test the proposed solution on other data sets such as the BookCrossing [21], AmazonReview [15], and MovieLens 1M [11] data set allowing us to make more constructive conclusions on the generalizability of the proposed solution.

Another aspect that can be explored in the future is how different similarity metrics and/or clustering algorithms affect the performance of the CTCF recommender system. The probability of injecting the ratings into the data set p is likely to also have an impact on the performance of the algorithm and the impact of this variable can also be tested.

It was observed that the optimal number of clusters for the CTCF-RS lies somewhere between 15 and 30. Since the intervals in the experiments were quite large it is hard to pinpoint an exact value for the optimal number of clusters, therefore, another possible extension to this research would be to conduct the experiments again with smaller intervals for the number of clusters. Similarly, the experiments showed that the accuracy of the recommender system was highest when the cutting point was 30 and the coverage was highest when the cutting point was 110 however cutting point values beyond that range were not experimented with and therefore can be considered for future extensions.

The current implementation of the CTCF-RS clusters the movies in the tail solely based on the genre of the movies which is provided in the data set. However a potential improvement to the algorithm would be to include other variables such as actors, release year, and directors to cluster the movies together. It can also be considered to cluster the movies based on derived variables such as the average rating, popularity, and likability as was done in the research by Park and Tuzhilin [17].

6 Conclusion

This paper introduces a modified version of the clustered tail recommendation method proposed by Park and Tuzhlilin [17], for collaborative filtering based recommender systems named CTCF-RS. The research question being investigated in this paper is *'To what extent can clustering be applied to the long-tail of collaborative filtering recommender systems such that more long-tail items are included in the set of recommendations while not affecting the accuracy of the recommender system and how do the number of clusters and the cutting point have an influence on this?'* Based on empirical results gathered through experimentation on the 100K MovieLens data set [11] it can be concluded that clustering can be applied to a collaborative filtering based recommender system to reduce the effects of the long tail recommendation problem (LTRP).

A negative correlation was observed between the cutting point value and the accuracy and diversity of the recommender system, i.e a lower cutting point value improved the accuracy of the predicted ratings and the diversity of the recommended items. The cutting point value of 30 performed best for the RMSE, MAE, and diversity whereas a value of 110 performed the worst. The contrary however was observed when the coverage of the recommended items was considered. A higher cutting point value resulted in an improvement in the coverage of the recommended items suggesting a positive correlation between the two parameters. A value of 110 gave the highest coverage whereas a value of 30 gave the lowest. The CTCF-RS outperformed the baseline CF recommender system in terms of RMSE and coverage regardless of the cutting point and the number of clusters, however, the CTCF-RS only outperformed the baseline CF recommender system in terms of the MAE and diversity when the cutting point value was lower than 70.

The optimal number of clusters that yields a high value for all performance metrics was found to be somewhere between 15 and 30 regardless of the value of the cutting point. It is clear that a value for the number of clusters lower than 15 is not good for the recommender system with respect to all performance metrics because a lower number of clusters forces movies that are not similar to be placed in the same cluster. This makes the injected ratings more inaccurate and thereby reducing the performance of the system.

The increase in coverage of the CTCF recommender system suggests that more long-tail items are included in the recommendation set, however, this comes at a cost of reduced accuracy and diversity of the recommended items. A tradeoff between the two values must therefore be made and an optimal value for the cutting point would have to be chosen that performs well at both ends of the spectrum.

Several directions for further extensions to this research have been identified. Firstly, the scalability of the algorithm can be addressed. The algorithm takes significantly longer than the baseline collaborative filtering recommender system and can therefore be improved. Secondly, the generalizability of the proposed solution can be investigated by conducting experiments on other data sets such as BookCrossing [21] or AmazonReview [15]. So far it has only been experimented with, on a single data set making it hard to make conclusions about the generalizability. Another extension that can be made to the algorithm would be to explore the effects of other hyper parameters of the algorithms such as the similarity metric, clustering algorithm, or the probability of injecting a rating p. Next, it can be considered to reduce the intervals of the number of clusters and the range of cutting point values can be extended beyond 30 and 110. Finally, the clustering algorithm can be extended to use other variables such as the release year, actors and directors of the movies, or even derived variables such as the average rating, popularity, and likability of a movie.

7 Responsible Research

According to the International Science Council "Scientists are responsible for conducting and communicating scientific work with integrity, respect, fairness, trustworthiness, and transparency" [1]. This section focuses on the responsible research practices followed while conducting this research. Section 7.1 discusses how the research follows the guiding principles of scientific integrity such as honesty, responsibility, and transparency. Section 7.2 demonstrates how the conducted experiments/research can be reproduced by all readers.

7.1 Scientific Integrity

This research uses information from several studies, articles, and journals that were relevant to the topic of recommender systems. To give credit to the author of those works and to avoid plagiarism all sources have been referenced using the IEEE citation style.

The MovieLens 100K [11] data set used in this research is a publicly available data set that is free to be used by all users given that certain conditions are met. This research meets all of the listed conditions and can therefore safely use the data set.

7.2 Reproducibility

Scientific research is considered to be reproducible if the reader can produce the same results as are claimed in the research. To ensure reproducibility of this research the used code base has been made available on a GitLab repository[2]. The code base has been documented thoroughly to ensure that readers are easily able to understand the written code. Additionally, the repository also contains a README.md that provides clear instructions on how to run the project and replicate the experiments if required.

[2]https://gitlab.com/yashmundhra/recommender-system-rp-final

In addition to the code base, the methodology in section 3 and experimental setup in section 4.1 provide an extremely detailed description of how the experiments have been conducted and what parameters were used. Together with the code base, and the description any reader should be able to conduct the same experiments as conducted in this research.

It is important to note that it is extremely likely that the findings from a reproduced experiment may not be the exact same as the ones presented in this research. The reason for this is that the algorithm uses a clustering algorithm that assigns items to multiple clusters. The clustering is different in every iteration of the experiment.

References

[1] Responsible science. *International Science Council*, Apr 2021.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 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, 2005.

[3] Ajay Agarwal, Minakshi Chauhan, and Ghaziabad. Similarity measures used in recommender systems : A study. 2017.

[4] Abhinav Ajitsaria. Build a recommendation engine with collaborative filtering, Jun 2021.

[5] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. *Proc First Workshop High Performance Data Mining*, 04 2000.

[6] Chrish Anderson. The long tail. *Wired*, Oct 2004.

[7] Mojdeh Bahadorpour, Behzad Soleimani Neysiani, and Mohammad H. Nadimi-Shahraki. Determining optimal number of neighbors in item-based knn collaborative filtering algorithm for learning preferences of new users. *Journal of Telecommunication*, 9:163–167, 07 2017.

[8] Bing Bai, Yushun Fan, Wei Tan, and Jia Zhang. Dltsr: A deep learning framework for recommendations of long-tail web services. *IEEE Transactions on Services Computing*, 13(1):73–85, 2020.

[9] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12, 11 2002.

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

[11] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[12] Joseph Johnson and Yiu-Kai Ng. Using tripartite graphs to make long tail recommendations. In *2017 8th International Conference on Information, Intelligence, Systems Applications (IISA)*, pages 1–6, 2017.

[13] Siyi Liu and Yujia Zheng. Long-tail session-based recommendation. *CoRR*, abs/2007.12329, 2020.

[14] Andrew Luke, Joseph Johnson, and Yiu-Kai Ng. Recommending long-tail items using extended tripartite graphs. pages 123–130, 11 2018.

[15] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics.

[16] Yoon-Joo Park. The adaptive clustering method for the long tail problem of recommender systems. *IEEE Trans. Knowl. Data Eng.*, 25(8):1904–1915, 2013.

[17] Yoon-Joo Park and Alexander Tuzhilin. The long tail of recommender systems and how to leverage it. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, page 11–18, New York, NY, USA, 2008. Association for Computing Machinery.

[18] Rama Syamala Sreepada and Bidyut Kr. Patra. Mitigating long tail effect in recommendations using few shot learning technique. *Expert Syst. Appl.*, 140, 2020.

[19] Daniel Valcarce, Alfonso Landin, Javier Parapar, and Álvaro Barreiro. Collaborative filtering embeddings for memory-based recommender systems. *Engineering Applications of Artificial Intelligence*, 85:347–356, 10 2019.

[20] Hongzhi Yin, Bin Cui, Jing Li, Junjie Yao, and Chen Chen. Challenging the long tail recommendation. *CoRR*, abs/1205.6700, 2012.

[21] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, page 22–32, New York, NY, USA, 2005. Association for Computing Machinery.

A Root Mean Squared Error Bar Graph

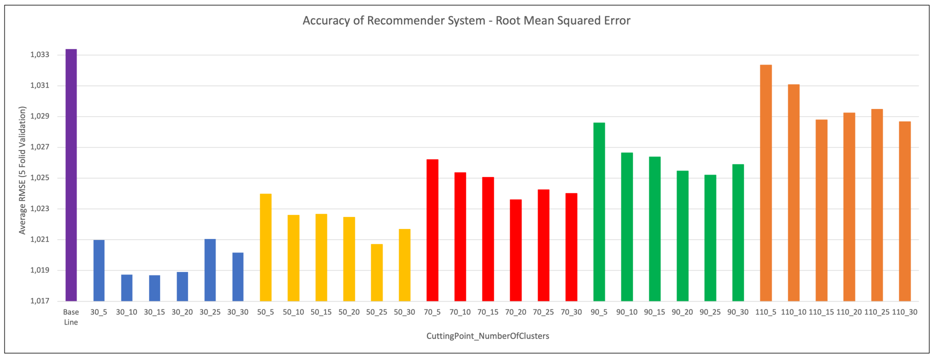

Figure 8: Root Mean Squared Error of recommender system with varying values for cutpoint and number of clusters. The bars are color coded based on the cutting point.

B Mean Absolute Error Bar Graph

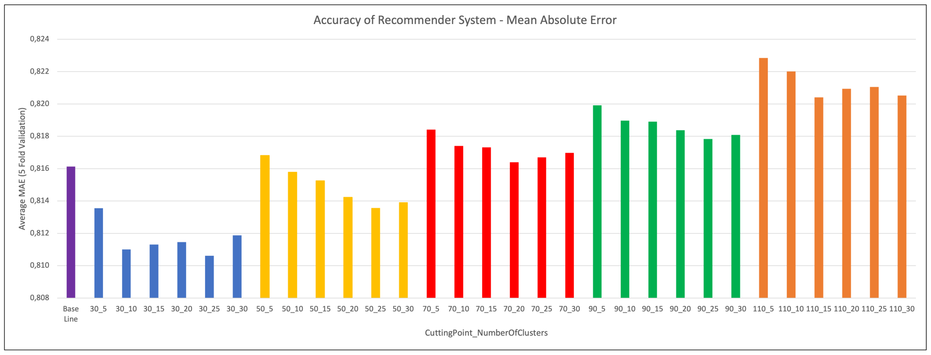

Figure 9: Mean Abosolute Error of recommender system with varying values for cutpoint and number of clusters. The bars are color coded based on the cutting point.

C Diversity Bar Graph

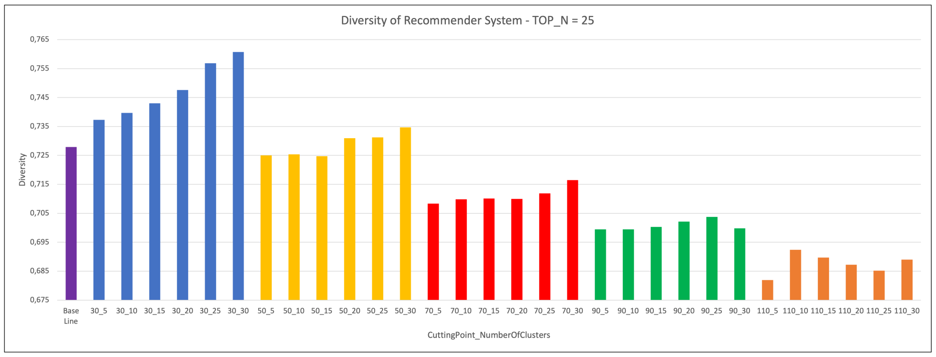

Figure 10: Diversity of recommender system with varying values for cutpoint and number of clusters. The bars are color coded based on the cutting point.

D Coverage Bar Graph

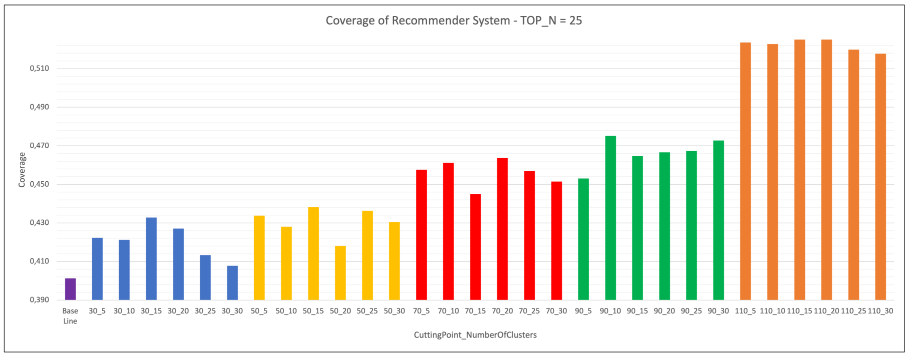

Figure 11: Coverage of recommender system with varying values for cutpoint and number of clusters. The bars are color coded based on the cutting point.