Spring 2018

USING FILTERS IN TIME-BASED MOVIE RECOMMENDER SYSTEMS

Ravee Khandagale
San Jose State University

USING FILTERS IN TIME-BASED MOVIE RECOMMENDER SYSTEMS

A Project

Presented To

The Faculty of Department of Computer Science

San José State University

In Partial Fulfilment

Of the Requirements for the Degree

Master of Computer Science

By

Ravee Khandagale

May 2018

SAN JOSÉ STATE UNIVERSITY

The Undersigned Thesis Committee Approves the Thesis Titled

USING FILTERS IN TIME-BASED MOVIE RECOMMENDER SYSTEMS

By
Ravee Khandagale

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

_____

Dr. Teng Moh, Department of Computer Science 05/02/2018

_____

Dr. Melody Moh, Department of Computer Science 05/02/2018

_____

Dr. Robert Chun, Department of Computer Science 05/02/2018

ABSTRACT

On a very high level, a movie recommendation system is one which uses data about the user, data about the movie and the ratings given by a user in order to generate predictions for the movies that the user will like. This prediction is further presented to the user as a recommendation. For example, Netflix uses a recommendation system to predict movies and generate favorable recommendations for users based on their profiles and the profiles of users similar to them. In user-based collaborative filtering algorithm, the movies rated highly by the similar users of a particular user are considered as recommendations to that user. But users' preferences vary with time, which often affects the efficacy of the recommendation, especially in a movie recommendation system. Because of the constant variation of the preferences, there has been research on using time of rating or watching the movie as a significant factor for recommendation. If time is considered as an attribute in the training phase of building a recommendation model, the model might get complex. Most of the research till now does this in the training phase, however, we study the effect of using time as a factor in the post training phase and study it further by applying a genre-based filtering mechanism on the system. Employing this in the post training phase reduces the complexity of the method and also reduces the number of irrelevant recommendations.

Keywords—Collaborative filtering (CF), target user, similar users, time-based, timestamp

ACKNOWLEDMENT

I thank my project advisor, Dr. Teng Moh for guiding, motivating and inspiring me not only through the project but through the entirety of graduate school. I also extend my appreciation to the committee members, Dr. Melody Moh and Dr. Robert Chun for their advice, suggestions and time on this project.

I also thank the department for its resources because of which I acquired the knowledge and skillset necessary to see this project to fruition.

I express a heartfelt gratitude towards my family and friends for their unconditional support and faith.

Table Of Contents

Table of Figures

Table of Tables

1. Introduction

Recommendation systems are used as part of analytics and services created for e-commerce, logistics and media websites or applications. They are designed in order to predict the kind of products a user may like and suggest such products to him or her. The user for whom the system generates recommendation is called the target user. Recommendation systems have to analyze a large amount of data in the form of users, products and their related information and devise an algorithm or behavior, called a model, which generates a customized list of recommendations for each user. There are systems which create a model according to their purpose and application. For example, a social media based recommendation system is modelled for giving a variety of recommendations rather than a mathematically accurate model. Essentially, recommendation systems are designed in order to provide custom recommendations for each user of the system. Their purpose is to filter content according to each user's taste and behavior. A few examples of websites using recommendation system are Amazon and Netflix. The systems used by these websites have proven to be effective to generate profits for these websites. According to the long tail effect, in contrast with products kept on shelf, recommendation systems help in generating profits because of their unlimited shelf space [10].

Typically, while generating recommendations, the system has to deal with a lot of data. Because of this, there is a need to devise algorithms which create a model which generates relevant and effective recommendations. Algorithms like collaborative filtering and content-based filtering are widely used by majority of recommendation systems. Collaborative filtering algorithms find similar users to the target user. To elaborate on this idea, say, in a movie recommendation system, a user named Alice likes a movie called *Insidious* which is a horror

movie. Now, say another user named Bob likes this movie *Insidious*. In general, if Alice and Bob like a number of same movies or movies of the same genre, they are considered similar users. If the target user is Alice, that is, if the system wants to generate recommendations for Alice, it will recommend the movies rated highly by her similar user, Bob. If Bob rates the movie *Silence of the Lambs* highly, the system will recommend to Alice the movie, *Silence of the lambs*. In content-based Filtering, on the other hand, the system finds similar items to the items the target user has rated highly. Continuing the previous example, the system recommends to Alice the movie *The Conjuring* because she likes the movie *Insidious* which is a horror movie and *The Conjuring* is also a horror movie. Here, *Insidious* and *Conjuring* are similar movies or similar items. Many systems also use a hybrid approach in order to generate recommendations. Hybrid systems are a combination of content-based filtering and collaborative filtering. Here, the system gives recommendations based on both similar users and similar items.

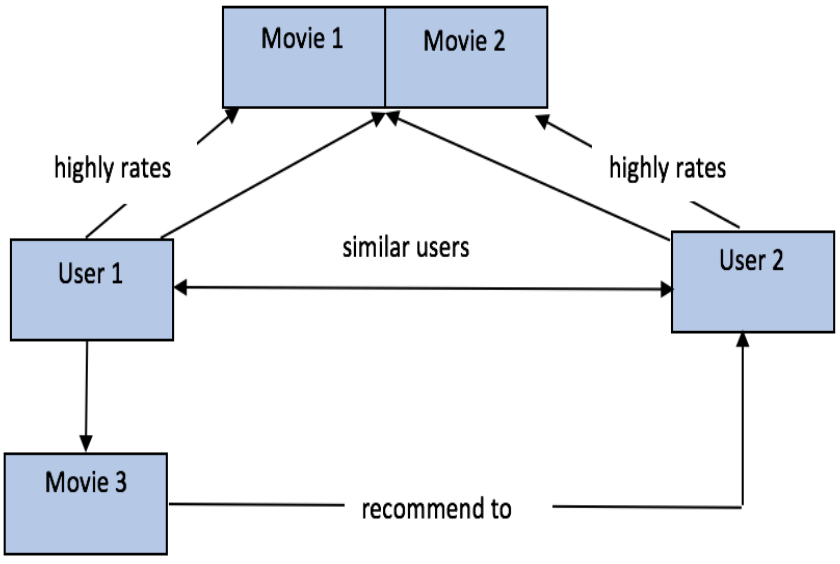

Fig. 1. An illustration of Collaborative Filtering

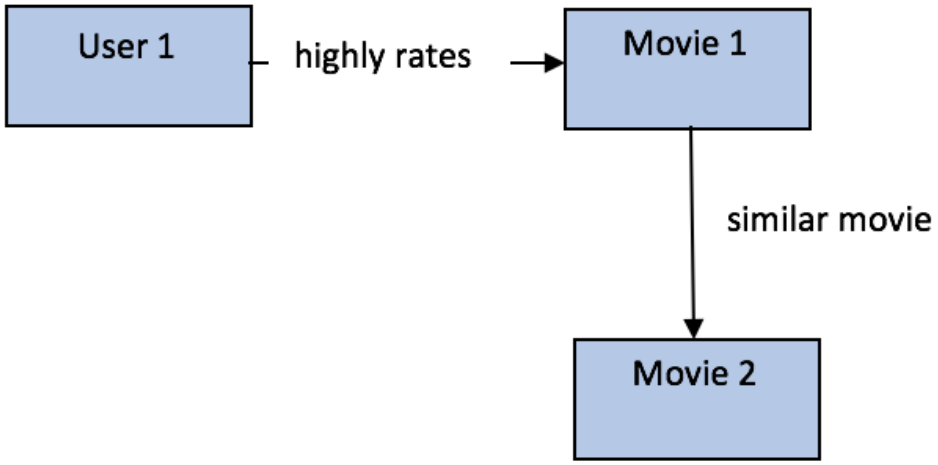

Fig. 2. An illustration of Content-based Filtering

Many commercial systems use collaborative filtering as the primary algorithm for designing their recommendation system. Collaborative filtering can be further categorized into user-based and item-based. By using a similarity measure to calculate how similar one user is to another user, user-based collaborative filtering is designed. The items rated highly by each user can then be used as a recommendation to the user's nearest neighbor. But, in a system with a lot of items and only a few ratings, user-user similarity can become ineffective. Additionally, due to changing preferences of users, the model needs to be recomputed. For this reason, item-based collaborating filtering was introduced. Item-based collaborative filtering is designed by calculating how close each item is to another. Typically, two items are considered similar if a lot of common users highly rate both the items. After calculating similarity between users or items, *k* nearest neighbors are found for each item or user in order to take their preferences as recommendations. Here, *k* is a number which is chosen by validation of the recommendation

model designed. That is, the value of *k* for which the model gives the highest precision, recall, accuracy or suitable measurement is chosen.

However, base algorithms like collaborative filtering or content-based filtering do not consider time as a factor in their design. Practically, time is a factor which can change user's preferences. For example, in a movie recommendation system, if a user has been watching a majority of horror movies for the past two years, that user will be recommended horror movies. However, if the user switches his/her preference to comedies, the user's new preference can be recognized as comedies. But, due to omission of time as a factor in filtering recommendations, the system will continue to recommend the user horror movies because horror movies make a majority of his/her past preferences. Another example is if a user Alice recently likes a movie called *Zootopia* and say Bob who is her similar user likes *Finding Dory* after he has highly rated *Zootopia*. Both are animated movies and intuitively, the chances that even Alice will watch *Finding Dory* after *Zootopia* are high. This is an important motivation, especially in user-based collaborative filtering, for the incorporation of time as a factor in recommendation systems. Also, devising new techniques for filtering the recommendations will increase the relevancy of the recommendations. With this intuition, it can be said that recently rated movies or items can be considered the latest preference of users. Having said this, there can be a timeline established on which there are items sorted based on when they are rated. The target user's latest preferences are the items which fall in the latest period on the timeline. After a model is trained, the *k* nearest neighbors of the user are found. Further, as post training, these latest items can be searched for in the time-sorted list of movies of the *k* nearest neighbors. On finding these items at time *t*, selected items after the time *t* can be tagged as recommendations. This paper includes three

approaches for selecting items using a filter that uses genre as information for filtering relevant recommendations. This approach is an enhancement to the approach proposed in paper [1].

2. Related Work

There has been research on using time as a factor in designing recommendation systems [1][2][3][4][5]. Timestamps have been considered while determining user similarity [1][2][3][4][5]. A big reason why this has become a popular research area is due to the fact that as time goes by, new users enter the system. Moreover, with time, new items, in this case, movies are added in the system. The profile of the already existing users in terms of items or movies they have rated keeps on changing as a result of this continuous change in data. N. Lathia, S. Hailes, L. Capra and X. Amatriain have studied the growth rate of ratings within a span of 2243 days in the Netflix prize dataset [1][12]. According to the researchers, there has been continuous arrival of users and movies in the system.

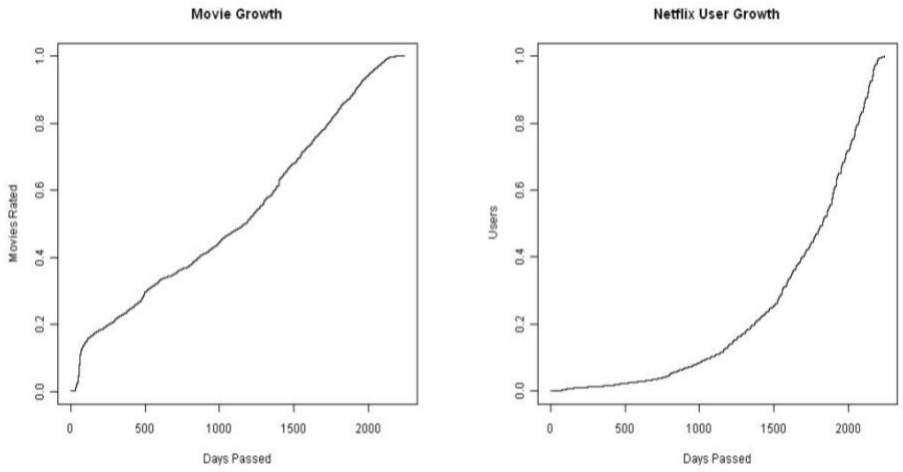

Fig. 3. Movie growth and user growth vs. Time in days from a survey conducted by Lathia, Hailes, Capra, Amatriain as illustrated in [2, Fig 1(a),(b)]

According to the paper, Figure 3 illustrates this best. It shows the growth of movies and users in a span of 2243 days. The researchers of this survey also noted that the summary values fluctuate over time which is an indication of how the overall distribution of ratings shifts as more users are active in the system. Similarly, Koren shows that the overall statistics of a system vary with time [11]. Furthermore, the researchers go on to state that because of the continuous influx of data and change in the rating content means that the training data that a model trained on will be different than the model it trained on previously [2]. These researchers showed that the traditional collaborative filtering algorithms produce low temporal diversity, that is, they recommended the same top-N items to users concluding that there was not much difference in the previous recommendations [2]. They also found out that users with large profiles are not recommended different items or movies [2]. They also designed and analyzed different methods of obtaining new or diverse recommendations without affecting the recommendation accuracy.

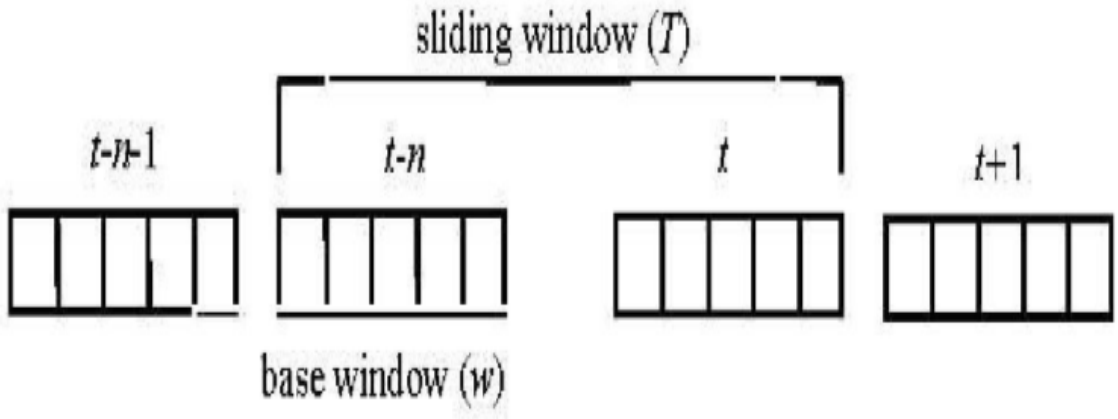

Fig. 4. Sliding window of length T in a time sorted list of movies as illustrated in [5]

Y. Shi also studied an approach to overcome the drawback of traditional collaborative filtering algorithms [5]. In that, she employed a sliding window technique in order to consider the user ratings which change over time [5]. Figure 4 shows the illustration of this sliding window of length T in a time sorted list of movies based on their timestamp.

These approaches include tweaking the training phase. However, this runs the risk of making the model unnecessarily complex just to get a few more or new recommendations.

The following research shows that time is used as a factor in recommendation system [6]. There has also been work done in developing a model to predict the probability of a user purchasing a product at a given time [6]. This paper makes use of time as a factor in a different way and takes into consideration the probability of purchase of product at that time [6]. Moreover, there has been research on a way to change an existing collaborative filtering algorithm based on product maturity in order to recommend items at a favorable time to a user [7]. Product maturity is directly related to the time it has spent in the system. In addition to this, there has been research on altering the movie-based collaborative filtering algorithm to a system which is based on timestamps of ratings in order to recommend to users the right movies at the right time [8]. All this research uses time in some way in order to alter or change the pre-existing or traditional collaborative filtering algorithm. While doing so, these algorithms include time as a factor in the training phase. This is the reason why the time to train these models increases. As a result, such algorithms also become complex.

Understanding this, L. Sun, E. Michael, S. Wang, Y. Li proposed a different approach of looking at the problem [1]. They proposed that instead of using time as a factor to train the collaborative filtering model, it could be included in the post training processing phase. Not only

did this approach make the training phase simple, it also made it easy to analyze and tweak the process to fit the needs of a system. More importantly, it also showed promising hit rate than that of the traditional recommendation system [1].

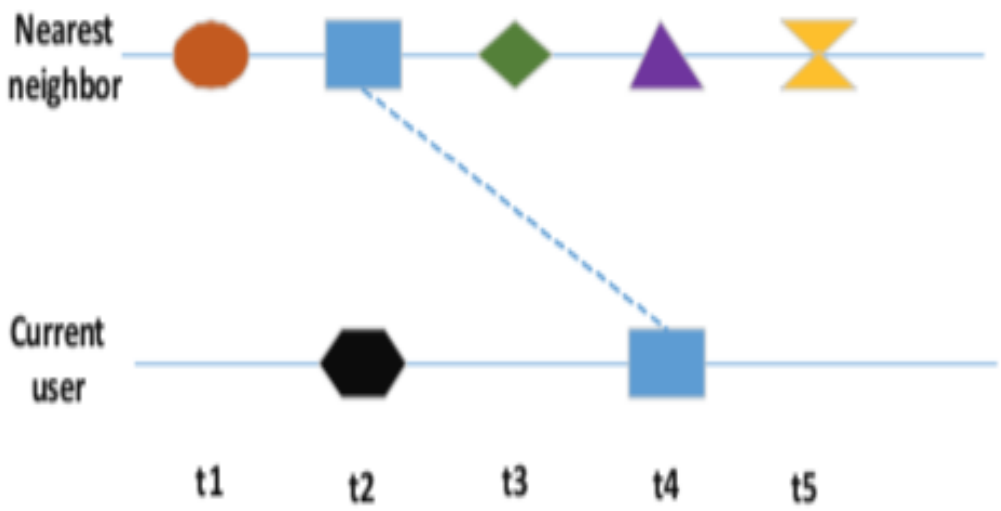

Fig. 5. Locating the latest item t4 in the nearest neighbor's time sequence as illustrated in [1, Fig. 1]

Initially, the model is trained by using the traditional collaborative filtering model in order to find the k-nearest neighbors for each user.

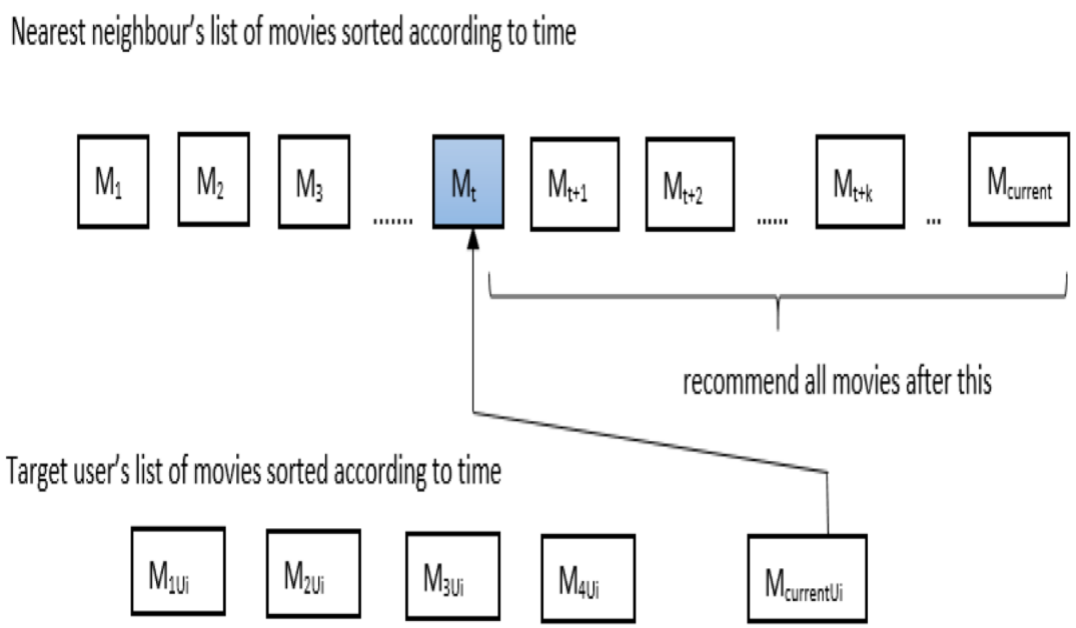

Fig. 6. Recommending all the movies after searching for the target user's latest item in the nearest neighbor's list

After this, the approach proposes a method which requires establishing a timeline of movies for each user. This timeline of movies is sorted in ascending order of the time at which they are rated. The latest movie on the timeline is chosen for the target user for whom the recommendations are to be generated. This latest movie is then searched in the k nearest neighbor's timeline. On finding this movie at time *t*, all movies after time *t* are recommended to the target user. This approach is simple, yet effective. As discussed before, this approach gives promising results [1]. However, it can be observed that this paper holds two assumptions. One assumption is that only the movies after time *t* are valid recommendations. This further goes to say that the movies before time *t* are not considered in the list of recommendations to give to the target user. As a result, even though the accuracy is said to be higher than the traditional collaborative filtering algorithms, employing this approach would mean losing out on other valid

recommendations. Another assumption is that all the movies after time *t* are potential valid recommendations. This entails that those movies after time *t* which would ideally not be good recommendations would also be included in the recommendation list. However, even though this by itself does not affect the other valid recommendations given by the system, it will decrease the conversion rate of movies a user will watch in practical e-commerce applications or websites. For this reason, there is scope for improvement by proposing an approach which filters out invalid recommendations and also considers relevant recommendations of the recommendations. This goes to say that the overall precision rate of the system will be increased. Moreover, like the approach proposed by Sun, Michael, Wang, Li, all this can also be done after the training phase of the recommendation system.

3. Proposed Methodology

There are two methods which are proposed which use time as a factor. These methods are an enhancement to the approach discussed in [1]. Both the methods need preprocessing of the same nature after the training phase before moving on to the actual implementation, post processing and analysis of each method.

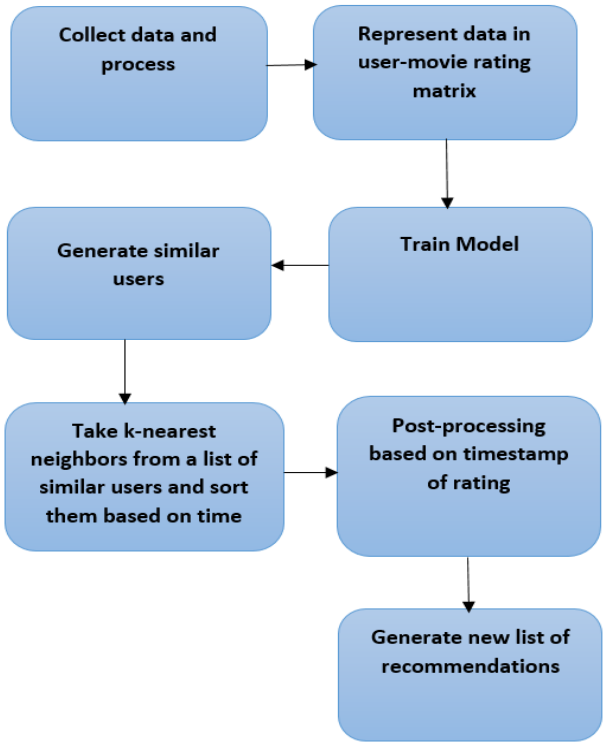

Fig. 7. Architecture Diagram

Fig.7 represents the high level block diagram of the entire process. Initially, the data is collected and represented in the form of a user-movie rating matrix. Using a traditional collaborative filtering model, model is trained. Similar users in the system are found after this. Each user's k-nearest neighbors are found using the similar users. After this, for the movies

watched by the target user and the user's k-nearest neighbors, a time-sorted list of movies is created. Post processing after the training phase, is done based on the timeline created and new recommendations are generated. The following sections define and explain each step of this process thoroughly

3.1 Data and its Representation

In any movie-based recommendation system which considers ratings, the initial data from analytics is in the form of user id, the movie id that the user has rated and the rating for that movie. This holds for all the users and the movies in the system.

TABLE I. USER-MOVIE RATING MATRIX

User/Movie	$Movie_1$	$Movie_2$	$Movie_3$	.	.	$Movie_m$
$User_1$	1					
$User_2$		1.5		2		2.5
$User_3$			5			
.	4					
.						3.5
$User_n$			3			

3.1.1 User-Movie Rating Matrix

To represent this, there is a rating matrix which is considered in order to solve the problem. In a movie recommendation system, if there are *m* movies and *n* users, then the rating matrix is *n x m*. In general, it is denoted by *R (n, m)* where r_{ij} denotes the rating given by the user *i* to the

movie *j*. This kind of setup however, does not account for the time at which the movie is rated. But, for this approach, there is a need to collect timestamps for each rating. In addition to this, for the post processing phase, we also need to consider the genre of each movie. Using this genre, we will calculate the top-n favorite genres of each user in the system.

3.1.2 Rating System

There are several factors a system or application can adopt in order to rate items. For example, Netflix uses likes and dislikes to rate their movies. This is a binary rating system whose rating matrix will have values either 0 or 1. 0 represents dislikes and 1 represents likes.

Most recommendation systems use a rating system which has rating 1, 2, 3, 4, 5. Some applications also use the scale from 1 to 10 where 1 represents the lowest rating and 10 represents the highest rating. The rating scale affects the similarity measure which is adopted while calculating the similarity between users. This is explained in the training phase section of this paper.

It is to be noted that in this paper, the ratings are considered from 1 to 5 where 1 represents the lowest rating and 5 represents the highest rating given to a movie. Floating point numbers for ratings are acceptable.

3.2 Training Phase

Using the rating matrix, the model is trained using the user-based collaborative filtering algorithm. The first step of this is to find the similarity between the users.

3.2.1 Finding Similarity

The goal here is to identify similar users to the target user. In order to do this, the similarity of the target user to the rest of the users in the system needs to be computed. Therefore, there is an intermediate problem statement of how to find and choose a function that calculates the similarity between two users. For this, an intuitive approach is to consider the number of high ratings given by two users to the same movies. There have been many functions studied for finding similarity. Methods like Jaccard similarity, Cosine similarity, Pearson correlation can be employed in order to find similarity. Most recommendation systems use Pearson similarity. However, deciding what kind of similarity to use also sometimes depends on what scale is used for rating. For example, for a binary rating system of likes and dislikes or 0 and 1, we can use the Jaccard similarity for calculating distance or similarity. However, for a rating scale of 1-5, Pearson correlation is found to work best [13]. In fact, on trying to employ Jaccard similarity for finding the nearest neighbors, we found that there was an increase in the false positive rate as compared to using the Pearson correlation. This is why, we decided to not pursue the Jaccard similarity as a similarity measure. The following equation gives Pearson similarity:

$$Pearson(u,v) = \frac{\sum k \epsilon I_u \cap I_v (r_{uk} - \mu_u).(r_{vk} - \mu_v)}{\sqrt{\sum k \epsilon I_u \cap I_v (r_{uk} - \mu_u)2} \sqrt{\sum k \epsilon I_u \cap I_v (r_{vk} - \mu_v)2}}$$

$\mathbf{I_u}$ are the items rated by user u,

$\mathbf{I_v}$ are the items rated by user v,

$\mathbf{r_{uk}}$ is the rating given by user u to movie k,

$\boldsymbol{\mu_u}$ is the mean rating of movies rated by user u

3.2.2 Training

We will employ Pearson correlation in order to find similarity between users. Then, the model is trained and the *k* nearest neighbors for each user are chosen from among the similar user list. In general, for every user u_i in the system, k nearest neighbors, K_{i1}, K_{i2}, … K_{ik} are calculated, where

$0 < i < n$,

K_{ik} – the kth nearest neighbor of user u_i

n – Total number of users

k can be chosen by experimental trying different values of *k*

3.3 Time based arrangement of movies

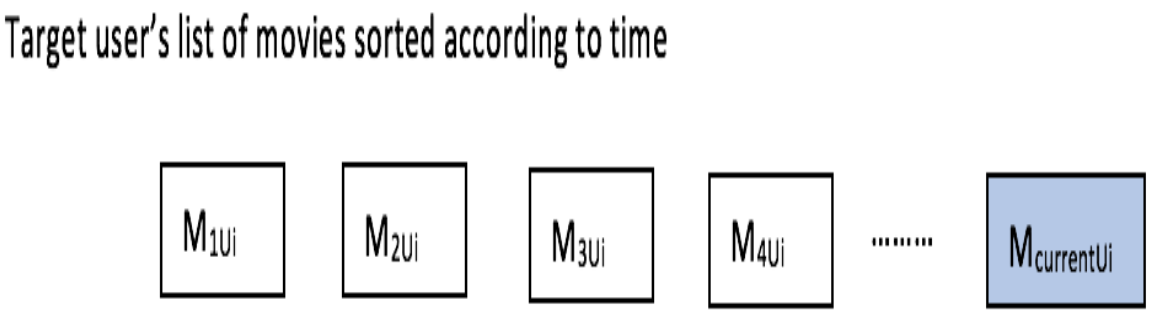

Fig. 8. Sorting movies for user U_i based on time for all 1 <= i <= n, where n is the total number of target users

For user *U*, a timeline can be established based on the time at which the user *U* has rated movies. The timeline T_U denotes the timeline of user U.

$\mathbf{T_U} = \{j_{t1}, j_{t2}, j_{t3}, \ldots, j_{tL}\}$ where,

$0 < j < m$,

m is the total number of movies,

$\mathbf{j_{tL}}$ is the movie at time t_L

L is the latest time on the timeline

3.4 Compute favorite genre of each user

Following are the steps for computing the favorite genre favGenre(u_i) of the target user. For each user u_i:

Create H (genre, count) where H is a hash table with keys as the genre and the value as the number of times the user likes a movie of that genre

For each movie m_j that u_i has rated, increment the value of the key, genre(m_j) in the hashmap H. Return top g favorite genres as favorite genres

TABLE II. GENRE-COUNT TABLE

Genre	Count
Adventure	9
Action	8
Crime	9
Comedy	2

3.5 Single-preference Approach - Filtering predictions based on favorite genre of the target user and the latest movie that the user has rated

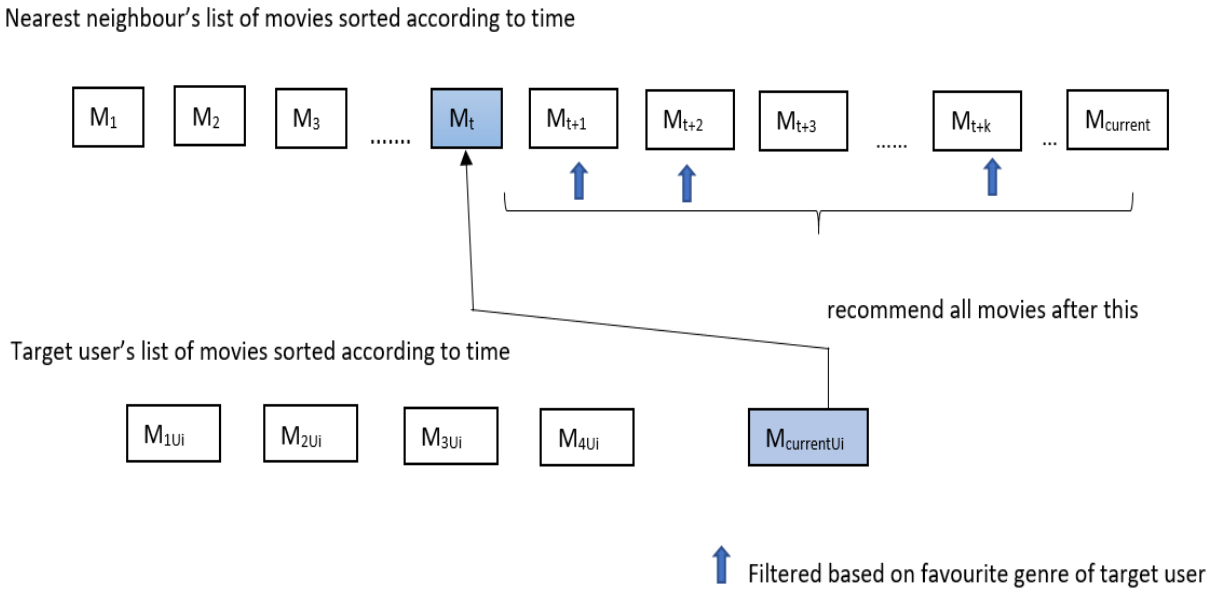

Fig. 9. Recommending filtered movies based on favorite genre after searching for the target user's latest item in the nearest neighbor's list

As illustrated by Fig.9 for each user u_i,

1) $T_{Ui} = \{j_{t1}, j_{t2}, j_{t3} \ldots j_{tL}\}$ denotes timeline of movies watched by the target user u_i and j_{tL} represents the last movie watched by u_i,

2) We obtain a nearest neighbor list for u_i: $K_{i1}, K_{i2}, \ldots, K_{ik}$.

3) For each neighbor K_{ik}, $T_{ik} = \{j_{ik1}, j_{ik2}, \ldots, j_{ikL}\}$

4) For nearest neighbor's list of movies, find movie j_{tL} in timeline of K_{ik},

5) Let t be the time in the timeline of K_{ik} at which movie j_{tL} is found. Now in the timeline of K_{ik},

6) For all times t` > t,

For movie $j_{ikt`}$, if (Genre($j_{ikt`}$) == favGenre(u_i) then, u_i.recommend ($j_{ikt`}$).

3.6 Multi-preference Approach- Filtering predictions based on favorite genre of the target user and the latest *m* movies the user has rated

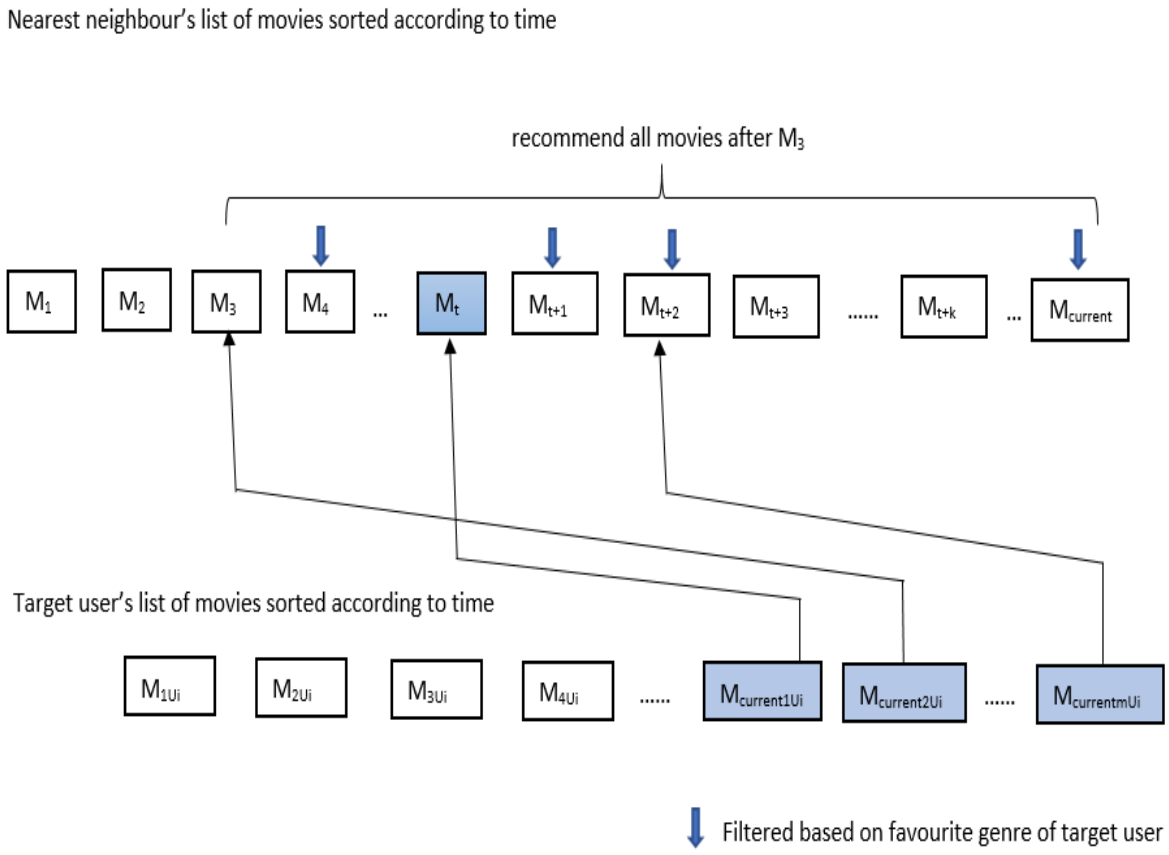

Fig. 10. Recommending filtered movies based on favorite genre after searching for the target user's *m* latest items in the nearest neighbor's list

This approach takes into account the latest *m* movies found on the timeline of the target user. This is different from the Single-Preference approach, where only the last movie was considered. Instead, we consider the last *m* movies, then locate the *m* movies on the nearest neighbor's

timeline and recommend movies after those *m* movies have been found. While doing so, we apply the genre filter like explained in the first approach.

As illustrated by Fig. 10. For each user u_i,

1) $T_U = \{j_{t1}, j_{t2}, j_{t3} \ldots j_{tL}\}$ denotes timeline of movies watched by the target user u_i and j_{tL} represents the last movie watched by u_i,

2) We obtain a nearest neighbor list for u_i: $K_{i1}, K_{i2}, \ldots, K_{ik}$.

3) For each neighbor K_{ik}, $T_{ik} = \{j_{ik1}, j_{ik2}, \ldots, j_{ikL}\}$

4) For nearest neighbor's list of movies, find movies $j_{tL1}, j_{tL2}, j_{tL3}, \ldots, j_{tLm}$ in timeline of K_{ik}

5) Let $t_1, t_2, t_3, \ldots, t_m$ be the times in the timeline of K_{ik} at which movie movies $j_{tL1}, j_{tL2}, j_{tL3}, \ldots, j_{tLm}$ are found. Now in the timeline of K_{ik},

6) For all times $t` > \min(t_1, t_2, t_3, \ldots, t_m)$,

For movie $j_{ikt`}$, if (genre($j_{ikt`}$) == favGenre(u_i) the, u_i, recommend ($j_{ikt`}$).

3.7 Two-sided Proximity Approach - Filtering predictions based on favorite genre of the target user and the proximity to the target user's latest movie found on the nearest neighbor's timeline

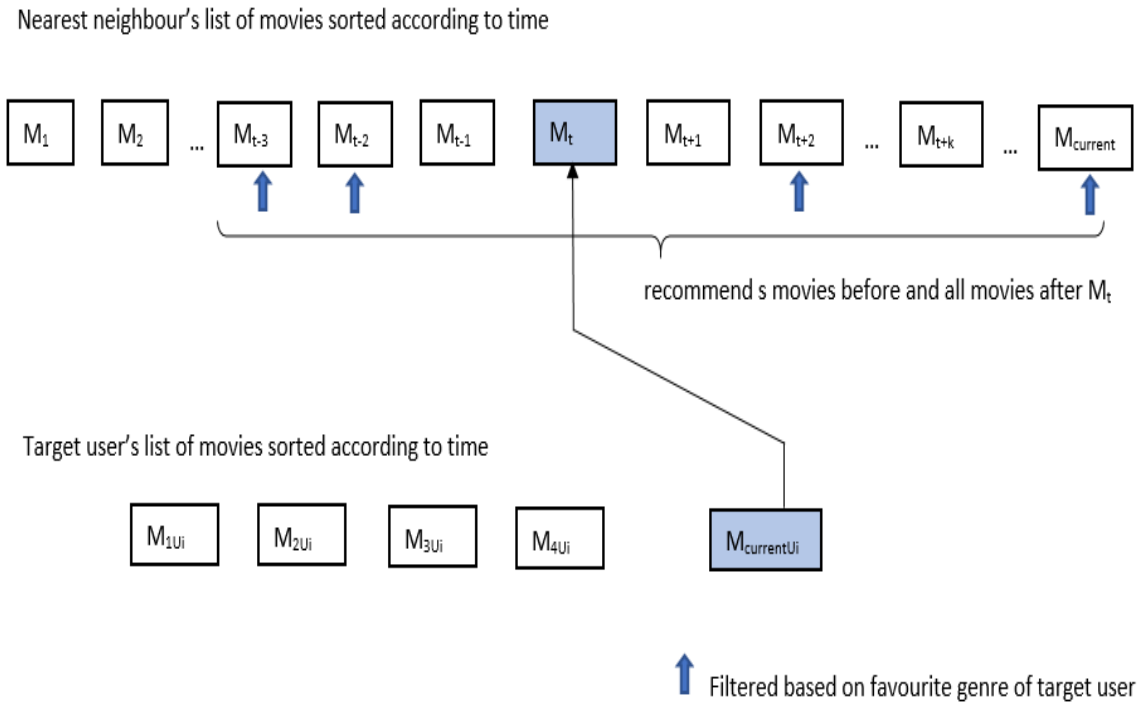

Fig. 11. Recommending genre-filtered movies which are before and after the location of the target user's latest item in the nearest neighbor's time-sorted list

The difference between this method and the Single-preference approach is that after locating the movie in the timeline of the nearest neighbor, instead of recommending movies after that movie is found, we recommend movies after as well as before that movie is found.

As illustrated by Fig.11, for each user u_i,

1) $T_U = \{j_{t1}, j_{t2}, j_{t3} \ldots j_{tL}\}$ denotes timeline of movies watched by the target user u_i and j_{tL} represents the last movie watched by u_i,

2) We obtain a nearest neighbor list for u_i: $K_{i1}, K_{i2}, \ldots, K_{ik}$.

3) For each neighbor K_{ik}, $T_{ik} = \{j_{ik1}, j_{ik2}, \ldots, j_{ikL}\}$

4) For nearest neighbor's list of movies, find movie j_{tL} in timeline of K_{ik},

5) Let t be the time in the timeline of K_{ik} at which movie j_{tL} is found. Now in the timeline of K_{ik},

6) For all times $s > t` > t_L$,

for movie $j_{ikt`}$, if (genre($j_{ikt`}$) == favGenre(u_i) the, u_i. recommend ($j_{ikt`}$).

4. Experiments and Results

This section presents a detailed experimental analysis of the proposed filtering method on time-based collaborative filtering approach. The results are compared to the time-based collaborative filtering approach proposed in paper [1].

4.1 Dataset

1. MovieLens Dataset with 100,000 ratings

The dataset has 100,00 ratings from 943 users. The number of movies they have rated is 1682 and the dataset is structured in a complete random fashion [1][9]. Users are numbered 1 to 943, and movies are numbered 1 to 1682, while ratings take values from 1 to 5 [1][9].

2. MovieLens Dataset with 1 million ratings:

The dataset has 1,000,209 ratings from around 6,040 users. The number of movies they have rated is around 3900 and the dataset is structured in a complete random fashion [1][9].

4.2 Experimental Setup and Evaluation Metrics

Initially, data was split into two sets, 80% was the training set and 20% was the testing set. The number of nearest neighbors to each user was taken after experimenting on different values of *k* as shown in the results section, which was around 100. Initially, the accuracy of the model was measured after the model was trained and validated using five-fold cross validation. This was compared to the accuracy of the traditional collaborative filtering model and that of the approach taken by paper [1]. Then, the precision of the model was calculated based on over 20,000 recommendations. A similar setup was run on 1 million dataset. The precision is compared to the approach taken by paper [1]. The results section explains in detail why precision was taken as a measure and why accuracy is not the only measure in the analysis.

4.3 Experimental Results and Analysis

The folds are given as r_1, r_2, r_3, r_4 and r_5. The Y-axis represents the number of ratings in the test data. Over an 80-20 split in training-testing data and K = 30, the following were the results on a five-fold cross validation:

TABLE III. ACCURACY USING FIVE FOLD CROSS VALIDATION

	r1	r2	r3	r4	r5
Sun-Michael-Wang-Li Approach	15394	13591	18491	16113	11189
Traditional Collaborative Filtering Approach	13922	14982	13987	13221	17221
Single-preference Approach	14927	13222	17891	15667	10982
Multi-preference Approach	14294	12453	17294	15342	12398
Two-sided Proximity Approach	13924	13753	17352	16203	12742

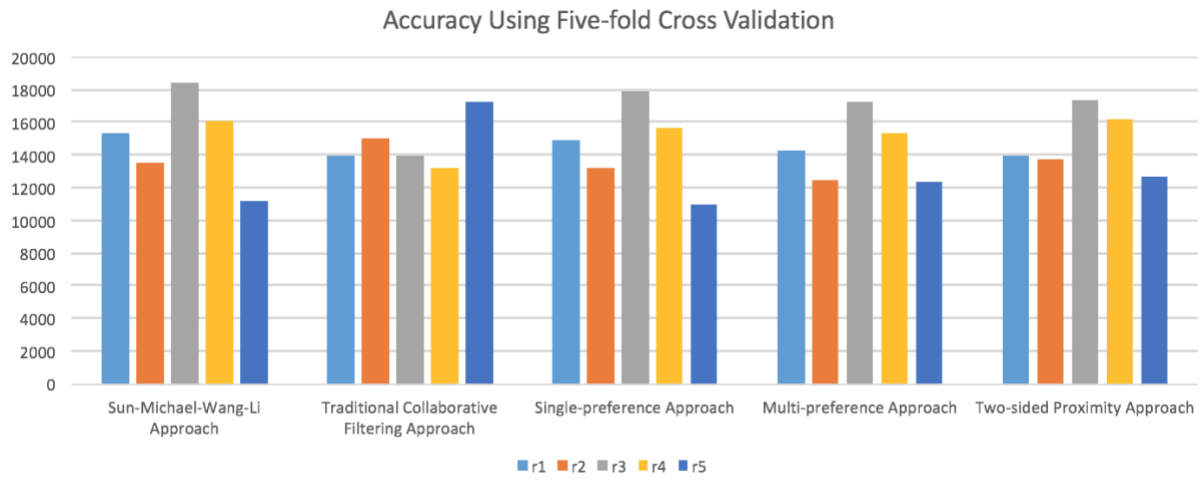

Fig. 12. Comparison of accuracy of proposed approaches with Sun-Michael-Wang-Li and traditional collaborative filtering method

4.3.1 Accuracy vs Precision

Here, the Sun-Michael-Wang-Li approach gives higher accuracy than the traditional collaborative filtering approach as well as the three approaches proposed in this paper. However, after examining the recommendations, we noticed that the false positive rate of the recommendations was high, that is, there were a lot of irrelevant recommendations in the approach proposed by Sun-Michael-Wang-Li. As compared to this, the accuracy of the approaches proposed in the paper is around 2.089% lower. This is because, for every *n* recommendation given by the Sun-Michael-Wang-Li approach, the filters proposed in this paper only pick a subset of those and give them as recommendations and as a result, this decreases the total number of recommendations. On the other hand, the false positive rate decreases and as shown in the rest of the paper, the precision increases as compared to the baseline.

Here, it is analyzed that accuracy is not the only correct measure of a system because accuracy measures number of recommendations over the total number of test points. However, in the approach proposed by Sun, Michael, Wang, Li, all the data points were not considered as relevant recommendations. There were many recommendations which were misclassified as relevant. In order to measure the degree of relevancy of the recommendations, we had to employ the measure of precision. The experimental results show that the precision of the Sun-Michael-Wang-Li approach is less in comparison to the methods defined in this paper.

Mathematically, accuracy and precision are represented as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

where,

TP = number of true positives

TN = number of true negatives

FP = number of false positives

FN = number of false negatives

4.3.2 Calculation of Precision On 100,000 MovieLens Dataset with the Three Approaches Compared to Baseline

TABLE IV: PRECISION % - 100,000 RATIINGS MOVIELENS DATASET

Method	Precision %	Post training time (ms)	Post training time – time required by filter (ms)	% Increase in Runtime compared to baseline
Baseline	85.99	618	-	-
Single-preference Approach	89.82	1278	781	106.7

Method	**Precision %**	**Post training time (ms)**	**Post training time – time required by filter (ms)**	**% Increase in Runtime compared to baseline**
Multi-preference Approach	91.12	2129	1632	244.49
Two-sided Proximity Approach (without filter)	87.00	900	-	45.63
Two-sided Proximity Approach	90.37	1809	1312	192.7

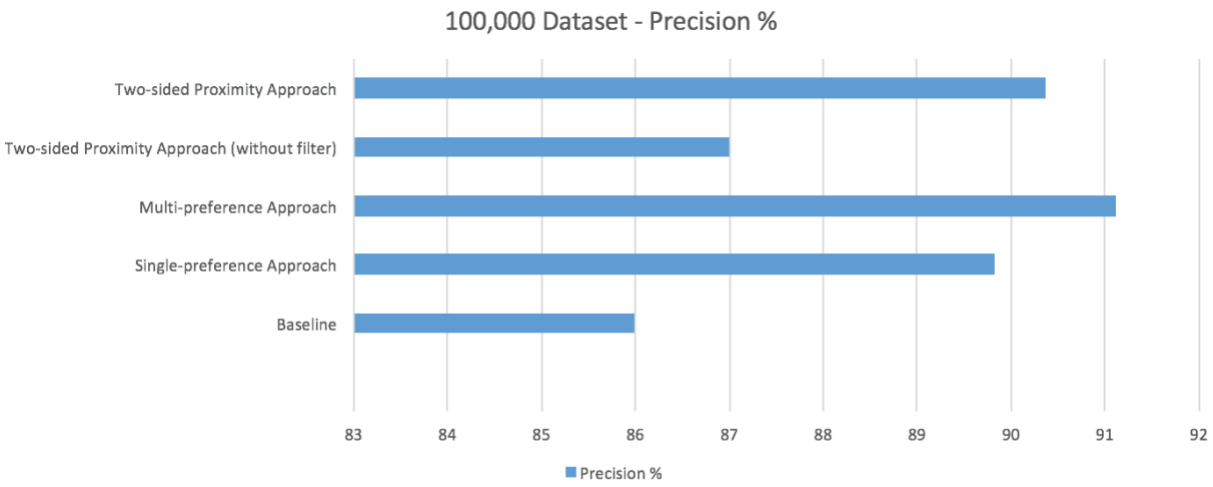

Fig. 13. Results of precision – 100,000 ratings MovieLens dataset

4.3.3 Calculation of Precision on 1M Ratings MovieLens Dataset with the Three Approaches Compared to Baseline

TABLE V: PRECISION % - 1 MILLION MOVIELENS DATASET

Method	Precision %	Total Post training time (ms)	Post training time – time required by approach	% Increase in Runtime compared to baseline
Baseline	84.93	83948	-	-
Single-preference Approach	94.12	90198	88907	6.9
Multi-preference Approach	93.39	89831	88540	7.00
Two-sided Proximity Approach (without filter)	85.28	103340	-	23.10
Two-sided proximity Approach	94.04	113921	112630	35.70

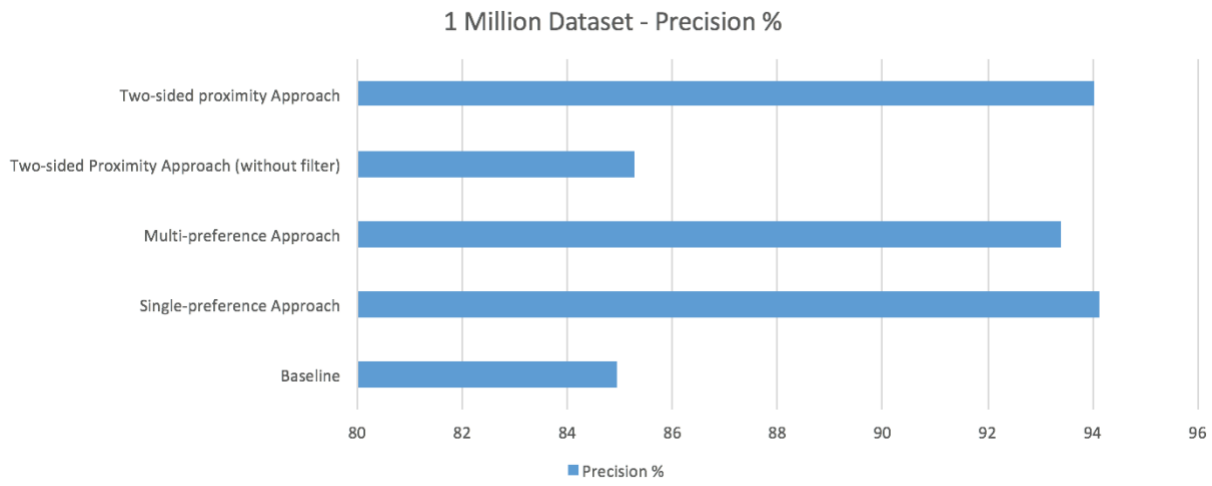

Fig. 14. Results of precision – 1M MovieLens dataset

4.3.4 Analysis

After analyzing the results, we can see that the proposed approaches perform much better than the baseline approach in terms of precision rate. After closely examining the results from the implementation, we found that the users who were recommended movies which were irrelevant for them, indeed had a much different genre preference than the movies recommended to them. This is a good indication that the idea behind using genre as a filter is a promising approach. The difference of around 10 % in precision rate would mean that in every 100,000 recommendations given to a user, the baseline approach recommends 10,000 more irrelevant movies than the approaches given in this paper.

The increase in precision rate has a tradeoff on the runtime. In the 100,000 dataset, the runtime of the three approaches is almost double that of the baseline. The genre filter used and the design of the approach is the main reason for increase in runtime. However, the filter is a one-time calculation. Therefore, as the dataset scales to 1 million, the increase in runtime in comparison to the baseline is not as much as it is for the 100,000 dataset.

5. Conclusion

There is a need to incorporate time as a factor in designing recommendation systems. Including time in the training phase runs the risk of complicating the model. This is why the post-training phase based on time proves to be effective in generating more relevant recommendations. The proposed method for filtering recommendations on time-based system proves to be effective in increasing the precision of the system. As a result, number of relevant recommendations increase. This is a promising experiment and can prove to be a baseline for future enhancements which can be done on this experiment. However, there are a few drawbacks to this approach as compared to the baseline. Due to the incorporation of the filter, the run time of the implementation increases as shown in the results. Moreover, implementing ensemble learning in the post training phase can also help the system make better decisions about recommendations at the cost of increasing the runtime. However, an increased precision rate would mean more meaningful and relevant movies will be recommended to the user. Practically, the system would not merely waste its recommendations by adopting these approaches and the users will see the suggestions they want to see. On that note, we can conclude that considering time as a factor and using filters is a good start in generating relevant recommendations.

References

[1] L. Sun, E. I. Michael, S. Wang and Y. Li, "A Time-Sensitive Collaborative Filtering Model in Recommendation Systems," *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, Chengdu, 2016, pp. 340-344.

[2] N. Lathia, S. Hailes, L. Capra and X. Amatriain, *Temporal Diversity in Recommender Systems[C]*, Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2010.

[3] G. Zhao, M. L.Lee, W. Hsu and W. Chen, *Increasing temporal diversity with purchase intervals*, Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2012.

[4] L. J.Jiao, S. L.Mei and W. Jiao, *Study on Session Recommendation Diversity in Web-based Recommendation System[J]*, Journal of Chinese Computer Systems. Vol.35,No.6, 2014.

[5] Y. Shi, *An improved collaborative filtering recommendation method based on timestamp*, 16th International Conference on Advanced Communication Technology. IEEE, 2014.

[6] J. Wang and Y. Zhang, *Opportunity model for e-commerce recommendation: right product; right time*, Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval. ACM, 2013.

[7] W. Hsu, M. L.Lee and K. Hu, *Utilizing users' tipping points in E- commerce Recommender systems*, Proceedings of the 2013 IEEE international conference on Data Engineering. ICDE, 2013.

[8] C. C.Su and P. J.Cheng, *Recommend at opportune moments*, Proceedings of the 7th Asia conference on Information Retrieval Symposium. AIRS, 2011.

[9] F. M. Harper J. A. Konstan, *The movielens datasets: History and context*, ACM Transactions on Interactive Intelligent Systems. vol. 5 no. 4 2015.

[10] C. Anderson, The Long Tail. Why the Future of Business Is Selling Less of More.NewYork,NY: Hyperion, 2006.

[11] Y. Koren. Collaborative Filtering With Temporal Dynamics. In ACM KDD, Paris, France, June 2009.

[12] Xavier Amatriain and Justin Basilico. 2012. Netflix Recommendations: Beyond the 5 stars (Part 2). Retrieved December 6, 2015 from http://techblog.netflix.com/2012/06/netflix-recommendations-beyond-5-stars.html

[13] L. Sheugh and S. H. Alizadeh, "A note on pearson correlation coefficient as a metric of similarity in recommender system," 2015 AI & Robotics (IRANOPEN),Qazvin,2015,pp.1-6.

Appendix

A.1 Choosing k for 943 users in the 100K MovieLens Dataset

TABLE A.VI. PRECISION FOR DIFFERENT VALUES OF K – 100,000 RATINGS

K	Precision - baseline	Precision – Single-preference Approach	Precision- Multi-preference Approach	Precision – Two-sided Proximity Approach
10	25.7	32.9	33.34	34.02
20	31.02	42.29	43.00	45.57
30	40.10	56.81	58.26	60.94
40	59.33	63.84	64.25	68.56
50	69.51	73.20	75.93	76.39
60	75.89	82.93	83.02	90.94
70	85.21	89.90	89.57	90.40
80	85.05	88.24	90.49	90.07
90	84.32	88.67	90.20	90.05
100	**85.99**	**89.82**	**91.12**	**90.37**

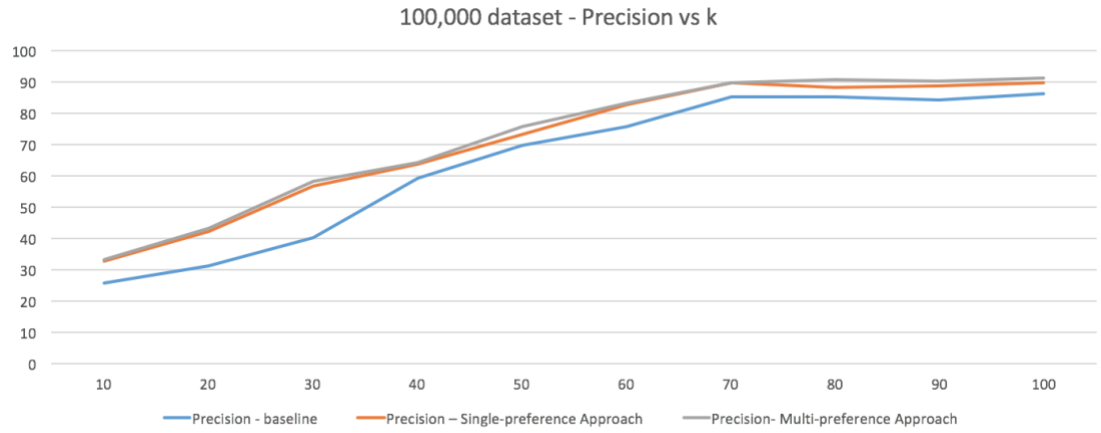

Fig. A.15 Results of precision for different values of k on 100K dataset

Fig.A.15 shows that the precision values all approaches converge at k = 70.

A.2 Choosing k for 6040 users in the 1 Million MovieLens Dataset

TABLE A.VII. PRECISION FOR DIFFERENT VALUES OF K - 1 M RATINGS

K	Precision - baseline	Precision – Single-preference Approach	Precision- Multi-preference Approach	Precision- Two-sided Proximity Approach
100	47.93	56.55	56.98	59.93
200	58.50	68.90	68.46	68.95
300	65.29	72.52	72.90	73.52
400	70.86	80.39	83.35	84.85
500	78.96	85.94	85.99	86.50
600	82.02	88.28	89.25	90.30
700	83.38	91.99	91.56	92.97
800	84.67	93.17	92.57	93.44
900	84.71	94.80	95.04	96.50
1000	**84.93**	**94.12**	**93.39**	**94.04**

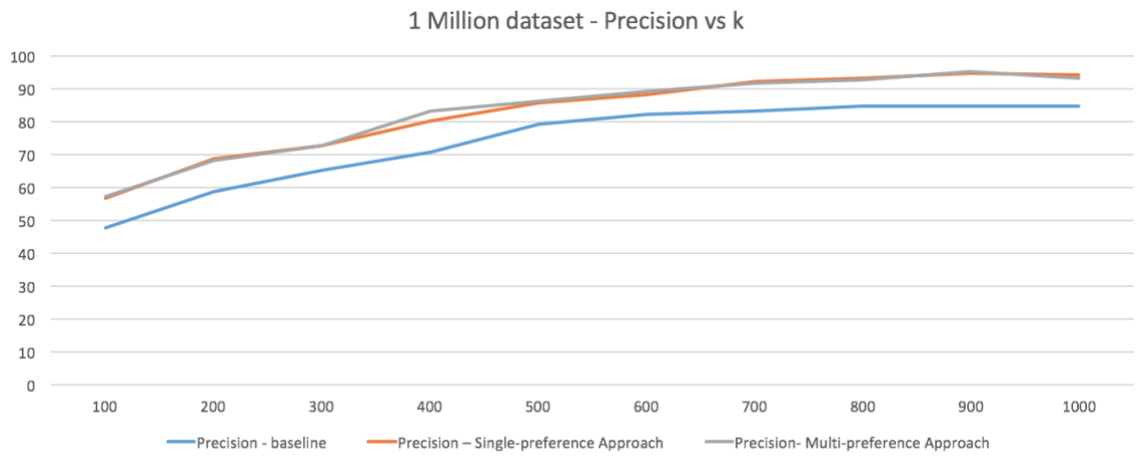

Fig. A.16. Results of precision for different values of k on 1 million dataset

Fig.A.16 shows that the precision values for all the approaches converge at k = 900

A.3 Choosing Threshold for High Rating

For the 1 million MovieLens dataset, we examined the results with two values for ratings which can be considered as threshold to decide recommendations. There was a considerable difference in the precision rate when threshold for rating was greater than 3 and when it was greater than 4. This makes the rating > 3 a good value for threshold. The high percentage of potential recommendations between the values 3 and 4, therefore, will yield more meaningful recommendations.

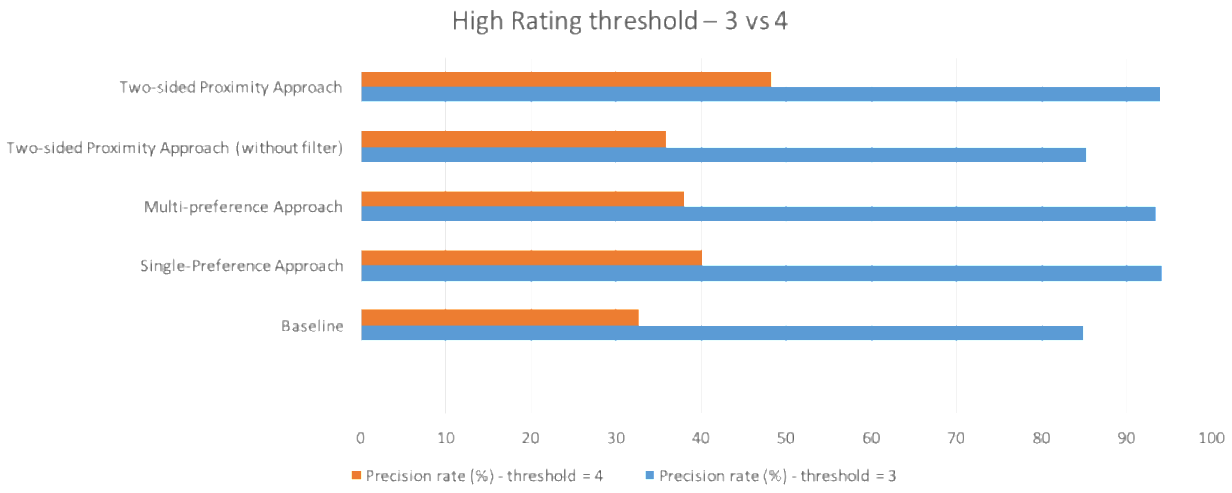

Fig. A.17. Results of precision for threshold values of high rating = 3 vs 4

TABLE A. VIII. THRESHOLD OF RATINGS – 3 VS 4

Method	Precision rate (%) (threshold = 3)	Precision rate (%) (threshold = 4)	Difference (the % of recommendations between 3 and 4 which can be recommended)
Baseline	84.93	32.65	37.2
Single-Preference Approach	94.12	40.08	54.04
Multi-preference Approach	93.39	38.06	55.33
Two-sided Proximity Approach (without filter)	85.28	35.93	40.44
Two-sided Proximity Approach	94.04	48.23	45.81