Time-Bin Based Neighborhood Selection for the Temporal Drift Phenomena using MovieLens and Netflix Case Studies

S. Aygun and M. Katipoglu

As the multitude of new applications arises, consumers get stuck amongst loads of choices. In the wake of solving this problem, the recommendation systems (RS) have been designed with different algorithms with a view to providing a decent amount of choice by filtering only the related ones to the user's interest. Even though the current dominating algorithms perform well, some of them are lack of detecting the temporal aspects of the user data such as the action frequency. The aim of this research is to present an algorithm that takes advantage of the temporal effects of the users. Time-bin based proposed algorithm is used to filter the best neighbors of an active user thereby performing well with respect to the traditional methods. The proposed algorithm has been extensively tested against different accuracy metrics on the Netflix Prize and Movie Lens Datasets

Introduction: In recent years, more and more applications started to provide an extensive amount of content. The abundance of choices that are provided by these applications to the users creates confusion and also makes it hard to find high-quality items for themselves. The aim of the recommendation systems is to provide the finest recommendations to each user as close as to their actual preferences. Recommender systems consist of a variety of methods. One of the most widely used approach is collaborative filtering (CF) [3]. It provides personalized recommendations with high accuracy. Common approaches to CF are latent factor models and neighborhood-based models. The latent factor models directly profile both users and products while neighborhood models analyze similarities between products or users. It is been made tremendous progress on the accuracy of CF with the Netflix Prize competition [4]. In general, Bell and Koren indicate that the lesson that has been learned from Netflix competition is that the aforementioned common approaches to CF target quite different levels of structure in the data, so these approaches are not optimal enough on their own [5].

In the search for underlying effects that alter the preference of users, the temporal drifting concept is observed. This phenomenon means that the rating scores of users on items tend to drift as the time passes [6]. Researches on the temporal dynamics, like [2], emphasizes that they have been faced with different kinds of time-related effects including complicated forms of concept drifts. Interconnected preferences of many users drift in different ways at different time points. Also, the temporal diversity of recommendations is an important factor that affects the quality of RS and provides feedback of the diverse tastes of users over time [7]. According to a review of temporal recommendation systems, it has been found that most of the current temporal approaches have focused on short-term and long-term temporal effects, but few of the approaches have focused on temporal drift and decay of CF [1]. Besides, Al-Hadi et al. point out in the review that the achievements of the current temporal recommendation systems are mostly based on matrix factorization giving results unsatisfactorily.

From the application perspective, one of the popular research subject from the CF is movie recommendation systems. With the presence of movie data sets, including user preferences for RS scientific research, such as MovieLens [11] and Netflix Prize [12], more and more studies have been attracted so far. In this letter, we do focus on the movie recommender systems and also contribute to the temporal analysis of user activity. This is to reveal the possible temporal drift of the user-in-test by retrieving the time information of the user rating activity. Thus, the time-bin of the user, which starts from the rating date of item-in-test and finishes at a time point up to then holding last ψ rating actions, is obtained. The same time-bin patches are extracted from the neighbor nominees, who are then sorted with some constraints. It is exposed to the temporal drift of user ratings simply by evaluating the temporal locality of the movie recommender systems. We provide an approach that finds the same-sized drifting patches, i.e., time-bins of different users, by taking advantage of same-like temporal rating activities. The proposed method addresses excluding negative temporal aspects, i.e., user inactivity time frame, of each user valid both for the active user-in-test and the prospective neighbors. In the scope of this letter, the RS framework for simulation purposes has been constructed in Python environment for each MovieLens and Netflix data set. Both conventional RS approach and time-bin based proposal are tested by measuring the comparative results using performance metrics given in the next section.

Background: In this background section, first similarity equation is given together with the prediction formula. Then, to compare the traditional RS architecture together with the own proposal, the popular metrics used in the literature are given in a tabloid way.

When making predictions with the neighborhood-based user-user CF methods, the most important concept is the measurement of correlations, i.e. in-between user similarity. There exists various similarity measures used in the RS like Pearson similarity, cosine similarity, Jaccard similarity [13], Spearman's rank correlation and Kendall's tau correlation, etc. Amongst all, the widely utilized similarity measurement is the Pearson Correlation (PC) [9].

Here in this research, we also use the Pearson correlation as given in Equation 1 where it is denoted with ρ_{uv}. The active user, namely user-in-test is denoted with u and prospective neighbors are given with v. Rating of user u on item i is shown with r_{ui}. Average rating for user u is μ_u and it is μ_u is for v. I_{uv} stands for movies rated by user u and v in common. In this work, we restrict the I_{uv}, as common ratings come only from the time-bins of having the same-like temporal drift. Also instead of all the ratings that belong to a user, we take the ratings that are found in the time-bin of interest which also means that if the target movie we want to predict is not in the neighbor's similar time-bin, then there would not be a way to use that time-bin during predictions. That is why all of the similar time-bins also needs to contain the target movie.

$$\rho_{uv} = \frac{\sum_{i \in I_{uv}} ((r_{ui} - \mu_u) * (r_{vi} - \mu_v))}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} * \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}} \tag{1}$$

After the correlation calculation, the up-coming important step is to decide the numbers of the neighbors which is denoted as k. In general any clustering method REFREF can be utilized, as k-nearest neighbors method fits well REFREF. The size of the k can be experimentally decided. In some experimental studies REFREF, it is fixed as we have set in this work by monitoring the performance both in traditional approach and in time-based proposal.

In user-user CF, after we get the nearest neighbors and thereby their correlations with the target user, we make the movie prediction using weighted average as shown in Equation 2. This prediction formula is also relevant to our prediction approach with time-bins. Instead of using the weights as the correlations in between users, we use the correlations in between time-bins as the weights.

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v=1}^{k} \rho_{uv} * (r_{vi} - \mu_v)}{\sum_{v=1}^{k} \rho_{uv}} \tag{2}$$

After the prediction calculation, the evaluation metrics should be utilized to measure the the algorithm effectiveness. In Table 1, the performance metrics that used in the experimental test phase of our research is given.

Table 1: Evaluation metrics used in the letter

RMSE	$\sqrt{\frac{1}{\lvert TestSet \rvert} \sum_{(u,i) \in TestSet} (\hat{r}_{ui} - r_{ui})^2}$
MCC	$\frac{TP*TN - FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$
F1	$\frac{2TP}{2TP+FP+FN}$
Specificity	$\frac{TN}{TN+FP}$
Recall	$\frac{TP}{TP+FN}$
Precision	$\frac{TP}{TP+FP}$
Markedness	$PPV + NPV - 1$
Informedness	$TPR + TNR - 1$
Balanced Accuracy	$\frac{TPR+TNR}{2}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$

In Table 1, metrics are given considering the confusion matrix elements, $true\ positive\ (TP)$, $false\ positive\ (FP)$, $true\ negative\ (TN)$ and $false\ negative\ (FN)$. One of the main focus of this research is to include other metrics apart from the $root\ mean\ square\ error\ (RMSE)$ only. When evaluating the metrics, we use every rating given as another class of a multi-class classification problem. In other words, we do utilize $the\ one\ against\ all$ approach [10] with multi-class evaluations. The TP is taken into account, for any n^{th} class denoted by C_n, as all the instances that are correctly classified fitting into the class C_n itself. Assuming $1 \leq n \leq m$ where m is the total number of classes, TN for this n^{th} class is all non-C_n instances these are not classified as C_n fitting into other classes $C_{\{1..m\}-n}$.

Proposed Approach: In this section, the proposed approach including the motivation will be introduced. Before giving the details of our proposed approach, we primarily present the data set in use to link the motivation and the data set characteristics. In this research, we use two data sets: the *Netflix Prize* and the *MovieLens*. Both of the data sets have some advantages and disadvantages in the sense of the information included. In Figure 1 we show that the ratings of the Netflix data set are distributes with a correlation to the timeline while the timestamps of the ratings in MovieLens differ. We consider this behaviour as a crucial point for the temporal analysis as the bulk-users act in same sense in the Netflix data set. Besides, there is an important difference between the chosen data sets. The Netflix data set contains 5 rating classes starting form 1 to 5, but the MovieLens data set has 10 different classes starting from 0.5 up until 5 with 0.5 increments that signifies the half-stars in the ratings. This observation leds us to use multi-class classification as an another motivation behind the choice of the data sets.

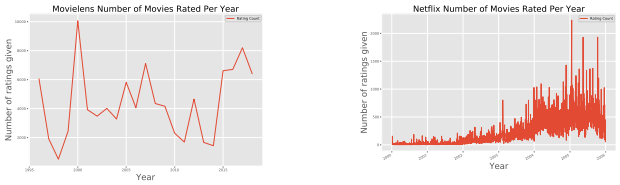


Fig. 1. Ratings Given Per Year

When it comes to the proposed methodology, the first step is to mention about the analysis of the user movie rating history in a different way. One of the main focus of the research is detecting the most appropriate time-bin in general. The time-bin size, taking the parameter of the rated movies, can be defined in the timeline starting from the rated movie in the test. Going back through the last N rating actions, the time-bin is decided dynamically. Decision of the N, which alters the time-bin size, is statistically handled by enumerating N, i.e., $N \leftarrow\ from\ 1\ to\ \epsilon$, where . At each item test, the active user time-bin size is parametrically tested to decide the best global N valid for the data set. On the way to this method, it had been primarily tried to create a global temporal drift by setting the static time-bins rather than taking the dynamic ones for each user. Static time intervals resulted in quite poor performance as well as not suiting well into the temporal drift phenomena that requires in-person analyses. Different users have different frequencies of giving ratings. Therefore, instead of fixing the time-bin size globally, the movie counts are statically analyzed which spans dynamic time intervals for each active user setting the edge of time-bin onto the item of test. In this work, bulk tests have been conducted to decide the N. It has been experimented with MovieLens and Netflix data sets we show the results in Figure 2. Considering th e variable N on the x-axis, RMSE performance of each data set is shown. We illustrate the results setting $10 \leq N \leq 50$ In Figure 2 a, Netflix gives lowest RMSE at N = 40, where as MovieLens gives at N = 10.

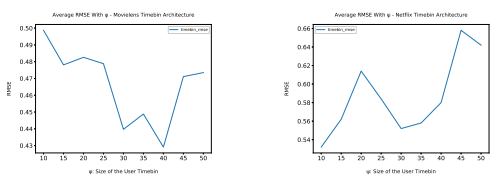


Fig. 2. Average RMSE as Timebin Size Changes

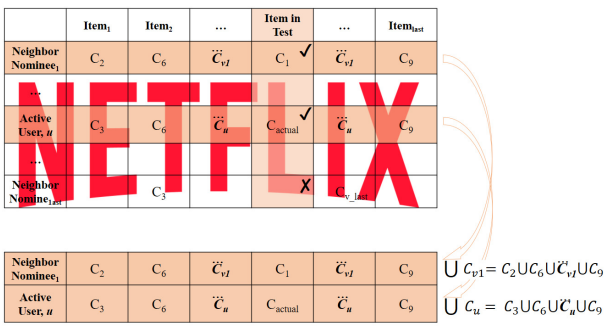
Next step in the proposed method is the decision of the best neighbors. After having calculated the similarity between active user and the prospective neighbors, the most similarly users are selected who are in the same temporal drift. At this point, we set a filtering to cross-check if the neighbour, v, satisfies some conditions or not. The user v has to have at least N amount of movies in its history. The test item has to be in its rated list. The timestamp of its time-bin should be the same starting & ending dates of the active user as the interval is dynamic.

The Timebin Based Nearest Neighborhood algorithm is about finding the most similarly drifting timebins using personalized timebin approach. The idea behind the personalized timebin view which we have formed after looking from the global perspective earlier and is also seen with the Temporal Dynamics research [2] as Koren says 'Every user drifts differently from every other user at different points in time'.

When detecting similar timebins, there exists some timebins where they dont have any different common movie ratings. Since we use the common movie ratings only when calculating correlation in between timebins we need to drop the duplicate timebins where they only have the same movies in common. That is after we found similar timebins, we look for the timebins that has ('neighbourId','nCommon', 'pearsonCorr', 'timebinI') tuples in common. If any two timebin has common neighbourId, the same number of common elements, the same pearson correlation and the timebins have the same starting index out of user's sorted watch history, then we drop one of them since they are duplicate.

Since we get the details of the Timebin Based Nearest Neighborhood, we are now ready to talk about how we make predictions by using the new similarity method. We first start by determining a user's movie rating that we would like to predict. We take a timebin of items ending with the target item which is not included in the timebin. Here we take timebin of 30 items since we have found it to be better at the global scale. Then we need to find the most similar timebins to the selected timebin. Since we will use the target movie rating and correlation of timebins, we only care about timebins which contains the target movie rating. That is why we first generate all possible timebins with the dataset we have and drop any timebin which does not have the target movie rating inside of it. Here it is important that the neighbour timebin size can take any value but the neighbour timebins should include some common movie ratings other than the target movie rating. That is why when we are evaluating, we have taken all the timebin sizes starting at 5 up until 100. Then, by using the common movie ratings found in the timebins, we calculate the pearson correlation of timebins. This correlation is the similarity of timebins which gives us how similar timebins act. We take this correlation and the ratings found in the timebins to the selected movie and calculate the weighted average given with the formula 2 as the prediction the to target movie rating of target user.

```
1:  procedure GET SIMILAR TIMEBINS(userId, timebin_i, timebin_size)
2:  similarTimebins ← list()
3:  timebin ← getTimebin(userId, timebin_i, timebin_size)
4:  pNeighbours ← getTimebinNeighboursWithTheMovie(timebin_i, timebin_size, movieId)
5:  for neighbourId in pNeighbours do
6:      neighbourTimebins ← getNeighbourTimebinsWithTheMovie(neighbourId, movieId)
7:      for timebin_neighbour, timebin_i, timebin_size in neighbourTimebins do
8:          n_common t ← getCommonElementCount(timebin, timebin_neighbour)
9:          if n_common <= minCommonTimebinElements then
10:             continue
11:         end if
12:         corr ← pearsonBetweenTimebins(timebin, timebin_neighbour)
13:         similarTimebins.append(neighbourId, corr, n_common, timebin_i, timebin_size))
14:     end for
15: end for
16: return similarTimebins
17: end procedure
```



Evaluation Results: When evaluating our Timebin Based Nearest Neighbourhood approach, we use multi class classification one against all approach [10]. The reason we examine the results at rating level of granularity is that the Netflix dataset and the Movielens dataset have different number of rating classes. Our evaluation on the new approach is quite extensive and links have been provided to the reader as an open source resource github repository at the acknowledgement section of this paper. At the repository we have given all the results inside of jupyter notebooks that we have created. Readers can experiment with the architecture that we have provided. Here we will be mentioning some of the metrics we have used, the others should be examined at the link.

Table 2: Average RMSE Values

	Movielens	Netflix
Pearson	0.93	0.85
Timebin	0.47	0.59

Here at Table 2, we see that the time-bin based neighborhood performs much better than the pearson. It is important to point out that since we take the subset of the Netflix where every user rated at least 40 movies, we have high pearson correlations in pearson correlation than its classical values REFREF.

Then, we compare the precision, recall and F1 measure in this order and the results of both of the datasets shows that the timebin architecture performs quite better than the pearson correlation. The Figures 3, 4 and 5 shows the respective plots.

At the table 2 we see that on average of the Netflix and the Movielens dataset, the pearson correlation have a average RMSE value of 0.89, and the Timebin Based Similarity has the average RMSe value of 0.53. This increase in RMSE values proves that the aim of the project which is taking advantage of the temporal drifts at the similariy measure is achieved.

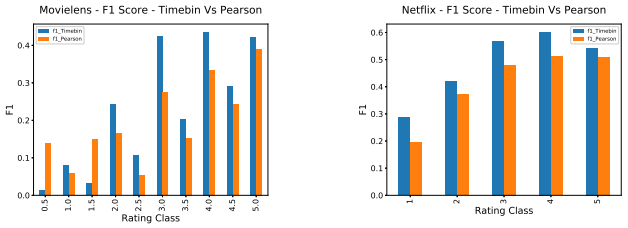


Fig. 3. F1 Score with Multi Class Rating Prediction

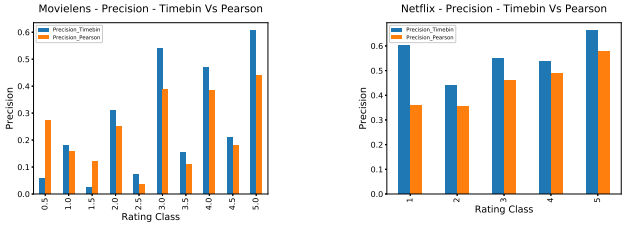


Fig. 4. Precision with Multi Class Rating Prediction

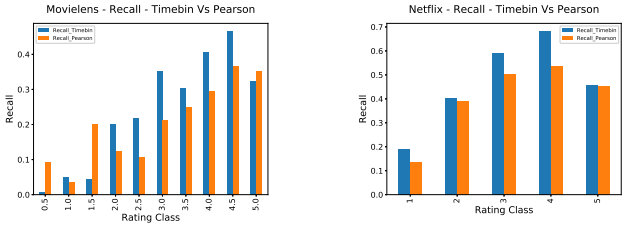


Fig. 5. Recall with Multi Class Rating Prediction

Conclusion: As we can clearly see from the evaluation results, newly proposed The Timebin Based Nearest Neighbourhood algorithm performs better than the pearson correlation in almost all metrics. Also the RMSE values is getting quite better which means this algorithm can be used as an alternative to the pearson correlation as well as the other recent algorithms.

To sum up, Timebin Based Nearest Neighbourhood is a similarity measure which performs well on both Movielens and Netflix dataset. This measure should be used whenever you want to take advantage of the temporal drift effects of users.

Acknowledgment: Temporal Drift Github Repository Link

S. Aygun and M. Katipoglu (*Yildiz Technical University, Computer Engineering Dept., Davutpasa, Istanbul, TURKEY*)

E-mail: mustafa.katipoglu@std.yildiz.edu.tr

References

1 Al-Hadi.: 'Review of the temporal recommendation system with matrix factorization', *I. Journal of Innovative Computing, Information and Control*, 2017, **5**, p. 1579–1594
2 Koren.: 'Collaborative Filtering with Temporal Dynamics', *ACM*, 2009, **9**, pp. 1406
3 Polatidis.: 'A multi-level collaborative filtering method that improves recommendations', *Expert Systems with Applications*, 2016, **10**, pp. 100-110
4 Koren.: 'Advances in collaborative filtering', *Recommender Systems Handbook, Second Edition*, 2015, **41**, pp. 77-118
5 Bell.: 'Lessons from the Netflix prize challenge', *ACM SIGKDD Explorations Newsletter*, 2007, **2**, pp. 75
6 Dror.: 'Yahoo! music,collaborative filtering,matrix factorization,recommender systems', *RecSys'11 - Proceedings of the 5th ACM Conference on Recommender Systems*, 2011, **7**, pp. 165-172
7 Lathia.: 'Temporal diversity in recommender systems', *SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010, **7**, pp. 210-217
8 Toscher.: 'Improved Neighborhood-Based Algorithms for Large-Scale Recommender Systems', *SIGIR 2010 Proceedings - 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2010, **7**, pp. 210-217
9 Toscher.: 'Similarity Measures used in Recommender Systems: A Study', *International Journal of Engineering Technology Science and Research*, 2017, **6**, pp. 619-625
10 Varutbangkul.: 'One Against All Approach', *Comment on the researchgate.net about one against all approach*
11 movielens.: 'Movie Lens Movie Dataset', *Movielens Link*
12 netflix.: 'Netflix Prize Movie Dataset', *Netflix Prize Link*
13 Sujoy B., Sri Krishna K. Manoj Kumar T.: 'Information Sciences, *International Journal of Engineering Technology Science and Research*, 2019, **11**, pp. 53-64