Temporal Phenomenon in Recommender Systems with MovieLens and Netflix Case Studies

XXXXX XXXXX XXXXX XXXXX

Abstract: Recommender systems are used in various applications to boost the prediction accuracy of user preferences. Users have actions and choices for items such as songs, products, and movies. The recent developments in recommendation frameworks support precise user decision on any item depending on the actions of logged user. Although the existing algorithms exhibit good performance, some temporal aspects of user data, such as action frequency, require attention. In this study, a new algorithm is introduced that uses the temporal effects of users. A time-bin-based algorithm with temporal features was employed to filter the best neighbours of an active user and it performed well over the traditional method. The proposed algorithm was extensively tested against different performance metrics using the *MovieLens* and *Netflix Prize* datasets. The accuracy metric demonstrated an improvement of approximately 2% when the proposed time-bin-based method was applied.

Keywords: collaborative filtering; interest drift; temporal features; temporal recommendation systems; time-bin-based similarity

1 INTRODUCTION

With the emergence of a multitude of new applications, consumers find themselves stuck amongst numerous choices. The abundance of options offered by these applications creates confusion and makes it difficult to find relevant items. Recommendation systems (RSs) aim to provide the finest suggestions for each user. Recent efforts have demonstrated the success of the currently used architectures; however, the inclusion of temporal actions from system users can be a tool that improves efficiency.

Collaborative filtering (CF) is a commonly used RS implementation technique in the literature [1]–[3]. From the application perspective, movie-based RS is a favourite research subject in CF [4]–[6]. Since the development of scientific movie datasets, such as MovieLens [7], [8] and Netflix Prize [9], more studies are investigating this research topic. Using the aggregation of other users and their preferences, the *collaboration* sources further information for item rating prediction. The selection of neighbours is determined using the similarity weights between users [10]; this selection procedure can be enhanced using prior knowledge regarding temporal actions. Users who interact with an application may alter their preferences over time; thus, the concept of *temporal drift* is observed [11]. Studies related to temporal dynamics demonstrated that different types of time-related effects can occur, including complicated forms of concept drift [12]–[14]. Individual user preferences and actions may decay or increase with time. The temporal diversity of recommendations is an essential factor that affects the RS efficiency [15]. A review related to temporal RS stated that the current studies focusing on temporal drift indicate short-term and long-term effects of temporal features [16].

In this study, we focused on movie RSs and contributed to the temporal analysis of user activity using the aforementioned scientific movie datasets. We revealed the possible temporal drift of a user-in-test by retrieving the time information depending on the rating activities of users. Accordingly, we obtained the time-bin of users, which starts from the rating date of item-in-test and finishes at a time up to when the last ε number of rating actions were held. Time-bin patches were also derived from neighbour nominees, i.e., other users, who were then sorted with some constraints, as explained via an algorithm in the following sections. By evaluating the temporal locality of movie-based RS, the variable time-bin patches for prospective neighbours were tested. Thus, we provide a method that determines same-sized and different-sized drifting patches, namely time-bins, of neighbouring users whose temporal rating activity is compared with the user-in-test. We monitored the proposed method using several performance metrics, such as accuracy, balanced accuracy, sensitivity, negative predictive value, and the Matthews correlation coefficient (MCC). A detailed analysis of each movie rating class is given in plots.

2 BACKGROUND

In this section, the conventional similarity equation is given with the prediction formula used in this study. Then, to compare the traditional method and our approach, the well-known performance metrics used in the literature are introduced.

When making predictions with neighbourhood-based user-user CF methods, the most crucial concept involves measuring correlations, i.e., in-between user similarities. There are various similarity measures used in RSs, such as Pearson similarity, cosine similarity, Jaccard similarity, Spearman's rank correlation, and Kendall's tau correlation [17]. Among these, the *Pearson Correlation Coefficient* (PCC) is a widely utilised similarity measurement [18].

In this research, we used PCC as given in Eq. (1), where it is denoted by ρ_{uv}. The active user, namely user-in-test, is denoted by u and the prospective neighbours are denoted by v. Moreover, r_{ui} denotes the rating of user u for item i; the average rating for user u and neighbour v is denoted by μ_u and μ_v, respectively. I_{uv} represents the commonly rated items by users u and v.

$$\rho_{uv} = \frac{\sum_{i \in I_{uv}} \left[(r_{ui} - \mu_u) \times (r_{vi} - \mu_v) \right]}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \mu_u)^2} \times \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \mu_v)^2}} \quad (1)$$

After the correlation calculation, the next important step involves deciding the number of neighbours, denoted by k. Generally, any clustering method can be utilised but the top-k nearest neighbour is a commonly used approach [19]. The neighbour nominees are sorted based on their correlation coefficients, and top-k of them are marked as neighbours. The size of k can be experimentally determined because the system becomes saturated after a certain threshold.

After obtaining the nearest neighbours in user-user CF and their correlations with the active user, predictions are made using the weighted average formula in Eq. (2).

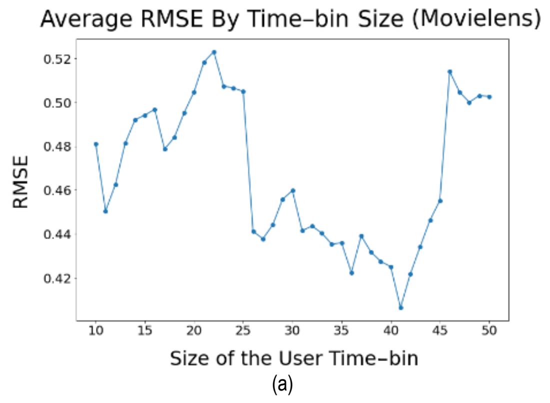
$$p_{ui} = \mu_u + \frac{\sum_{v=1}^{k} \rho_{uv} \times (r_{vi} - \mu_v)}{\sum_{v=1}^{k} \rho_{uv}} \quad (2)$$

Following the prediction calculation, evaluation metrics should be utilised to measure the effectiveness of the algorithm. Tab. 1 lists the performance metrics used in the experimental test phase of our research. *Root mean squared error* (RMSE) was utilised in global time-bin analyses, whose details are provided in the next section. *Accuracy*, *balanced accuracy*, *specificity*, and *negative predictive value* were employed for rating-based detailed monitoring at the end of this paper. Eventually, the compound metric, i.e., MCC, is visited for final remarks. The *compound* keyword here underlines the built-in structure of Matthews correlation because it holds the geometric mean of two sub-metrics, namely, *informedness* (INF) and *markedness* (MRK).

Table 1 Evaluation metrics used in this study

Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{\lvert Test\ Set \rvert} \sum_{(u,i) \in Test\ Set} (\hat{p}_{ui} - p_{ui})^2}$ $\hat{p}_{ui}$: actual rating value of *item* i from *user* u p_{ui}: predicted rating value of *item* i from *user* u
Accuracy	$\frac{Rating\ Matches}{TP + TN + FP + FN}$
Balanced Accuracy	$\frac{(TP\ /\ (TP\ +\ FN))\ + (TN\ /\ (TN + FP))}{2}$
Specificity	$\frac{TN}{TN + FP}$
Negative Predictive Value (NPV)	$\frac{TN}{TN + FN}$
Matthews Correlation	$\frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}$

TP: True Positives, TN: True Negatives, FP: False Positives, FN: False Negatives
$Matthews\ Correlation = \sqrt{INF \times MRK}$; $INF = \frac{TP}{TP+FN} + \frac{TN}{TN+FP} - 1$ and $MRK = \frac{TP}{TP+FP} + \frac{TN}{TN+FN} - 1$



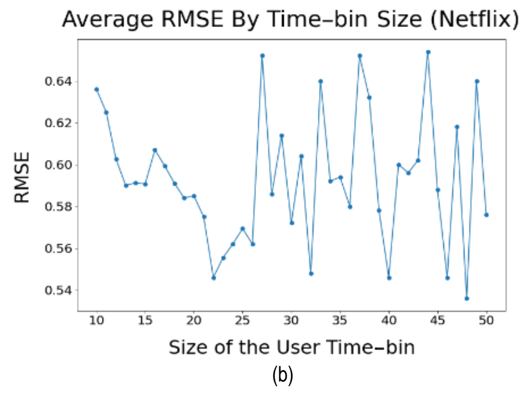


Figure 1 Average RMSE as the time-bin size changes for (a) MovieLens and (b) Netflix datasets

Tab. 1 presents metrics considering the confusion matrix elements, i.e., true positive (TP), false positive (FP), true negative (TN), and false negative (FN). When evaluating the metrics, we treat each rating prediction as a multi-class problem. In other words, we utilise the *one-rating-against-all-ratings* approach with multi-class evaluations [20]. Assuming $1 \leq n \leq m$, where m is the total number of rating classes, TP denotes the C_n classification of all C_n actual instances and TN denotes all non-C_n instances that are not predicted as C_n. Moreover, FP represents the C_n predictions despite belonging to non-C_n instances. Finally, FN denotes all C_n instances that are not classified as C_n. It should be noted that there is an essential difference between the datasets used in our analyses in terms of rating classes. The Netflix dataset contains five rating classes, starting from 1 to 5, while the MovieLens dataset has increments with 0,5 as half-stars in the ratings. For this essential difference, we considered rating-wise observations for the predictions.

3 PROPOSED APPROACH

In this section, the proposed approach is introduced. The first step involves analysing user rating history throughout the datasets. Therefore, it is important to detect the most globally suitable time-bin over the dataset. This pre-analysis technique attempts to find the most optimal time-bin size that is valid globally for all users. The optimality is determined by the RMSE induced by the time-bin size, as shown in Fig. 1. The time-bin size, taking the number of rated movies as a parameter, can be defined throughout a user timeline. A time-bin is determined by starting after the rated item-in-test and going back to the last N rating actions. The decision of N is statistically handled by an enumeration, i.e., $N \leftarrow \{1 \dots \varepsilon\}$. For each item, the active user time-bin size is parametrically tested to contribute the best global N. This generalised temporal phenomenon, N, depends on the dataset and pre-analysis via ε parameter.

In this work, bulk tests were performed to decide the static N value. With preliminary analyses using MovieLens and Netflix datasets, we showed the parametric tuning of N from 10 to ε, which lies in the x-axes of plots given by Fig. 1. Each parametric time-bin size for an active user is used for prediction. The predicted rating value based on neighbours with top-k PCC values is checked using RMSE and recorded. All possible time-bin sizes for N are set from 10 to 50 ($N \leftarrow \{10 \dots \varepsilon = 50\}$). Each recorded prediction attempt is used for the calculation of average RMSE from the exact time-bin size. In Fig. 1, the lowest RMSE value for static N at each dataset is separately investigated. In Fig. 1 (a), MovieLens yields the lowest RMSE at $N_{MovieLens} = 41$, whereas Netflix yields a value of $N_{Netflix} = 48$ in Fig. 1 (b). These static values are used for each user-in-test, i.e. active user, during the experiments.

Unlike prior efforts, we define a general time-bin of users in the datasets by restricting overall rating actions into the recent ones. We use an updated version of ρ_{uv} equation by I_{uv}, which now defines the common ratings only derived from the time-bins of test user and neighbour. Instead of all user ratings, we consider those included in the time-bin of users. The most recent rating actions performed before the test item are captured for the test user based on the static time-bin size. Then, similarities between the test user and its prospective neighbours are calculated. The time-bin calculation for neighbours is dynamically performed differently than the static time-bin of test user. The active user time-bin is set statically but the prospective neighbour time-bin size is flexibly in the range of 5 and 100 movie ratings from the whole user history. We observe that if neighbour selection is restricted to a single time-bin size, then the commonly rated item set, I_{uv}, cannot provide efficient information. A neighbour is expected to rate at least 5 items; otherwise, we monitor a significant accuracy drop in both datasets. Additionally, more than 100 item-sized time-bins can create a run-time overhead. All variable-sized time-bins are utilised for the best similarity score. However, there are several constraints for users to be considered as neighbours. If a target movie we want to predict is not in the time-bin of any prospective neighbour, then we cannot collaborate with this prospective neighbour. The test item must be in the rated list of the prospective neighbour, plus there must be some other rated items that are in common with those from the test user. Therefore, time-bins containing the target movie are primarily derived for collaboration with the test user. In the beginning, we define filtering to verify if the prospective neighbour satisfies the aforementioned conditions.

Fig. 2 depicts the initial selection of neighbour nominees based on the constraints. Let the table rows indicate any time-bins of users. An active user u and test item i with a rating value of C_{actual} are first considered. Then, any prospective neighbour, such as the first user in Fig. 2, is selected for the time-bin-based neighbourhood algorithm. For instance, *neighbour nominee₁* has a rating for the *item-in-test*, along with *item₁*, *item₂*, *item$_{last}$*, and some others whose ratings lie in vector $\dddot{I_{v1}}$. This nominee is suitable for the time-bin-based neighbouring because it contains the test item and some other items commonly rated by the active user. The intersection set of time-bin history, $I_{uv1} = I_u \cap I_{v1}$, is the focal point that returns feedback regarding the resemblance between two users. In contrast, *neighbour nominee$_{last}$* is never selected because the test item has not been rated. Our time-bin-based approach puts a temporal restriction on this neighbour selection procedure by reconfiguring user rating history, I. Accordingly, we compose the neighbour list based on common user preferences.

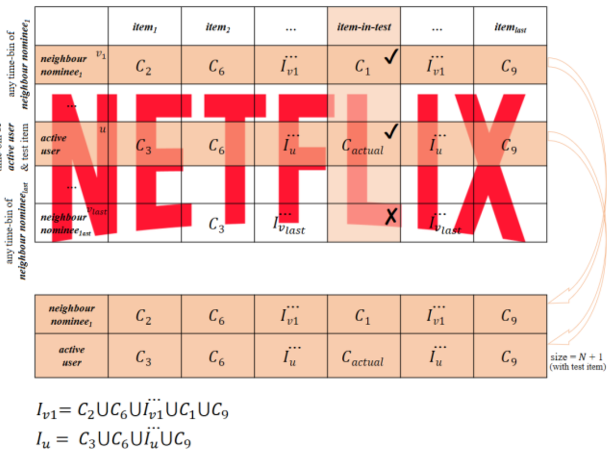


Figure 2 Example of neighbour nominee selection and rejection

During the neighbour selection procedure in our proposed scheme, PCC is calculated for each nominee. Because there can be multiple time-bins for each prospective neighbour owing to the flexible size, the selection of best performing time-bin is decided via the co-rated item count and PCC. Therefore, in addition to sorting significant PCC scores, the commonly rated item count is also considered. Finally, taking the sorted correlation weights, k actual neighbours are selected with their highly populated time-bins regarding common ratings concerning the active user.

Lastly, the prediction formula given by Eq. (2) is executed as being implicitly subject to the time-bins of users. The prediction formula treats neighbours selected with higher PCC values and commonly rated item counts. Accordingly, we focus on the correlations between the time-bins of neighbour and test user during the prediction. The inactivity time frames of neighbouring users are not included, thanks to the time-bin approach for better collaboration. In *Algorithm 1*, the overall steps are summarised.

Algorithm 1: Time-bin-based neighbourhood

```
1:  procedure SELECT NEIGHBOURS (testUserId, testMovieId, N)
2:      testUserTimebin ← getTimebin(testUserId, testMovieId, N)
3:                                  ▷ Based on N, test user time-bin is decided
4:                                  ▷ The most recent items are selected related to the test item rating date
5:      prospectiveNeighbours ← getProspectiveNeighbours(testMovieId, testUserTimebin)
6:                                  ▷ Filtering time-bins that include the test item and some other commonly rated items with respect to test user
7:                                  ▷ All possible variable-sized time-bins are inferred
8:      for all Neighbour in prospectiveNeighbours do
9:          [neighbourTimebins, PCC] ← getNeighbourTimebins(testUserTimebin, testMovieId, Neighbour)
10:         neighbourTimebins ← SORT(neighbourTimebins, PCC)
11:                                 ▷ Time-bins of any neighbour sorted by PCC values obtained with respect to the test user time-bin
12:         for all Timebin in neighbourTimebins do
13:             $n_{common}^{Timebin}$ ← getCommonCount( Timebin, testUserTimebin)    ▷ Common rating count
14:         end for  ▷ Iterating for all possible time-bins of a neighbour
15:         $n_{common}^{Timebin}$ ← SORT($n_{common}^{Timebin}$)
16:         finalTimebin ← selectBestPerforming(neighbourTimebins, PCC, $n_{common}^{Timebin}$)
17:                                 ▷ the one having higher ranking scores both in PCC and common rating count
18:     end for  ▷ Iterating for all possible neighbours
19:     return top-k distinct neighbours
20:                                 ▷ With the top PCC values between the selected time-bin patch of neighbour and the test user
21: end procedure
```

In *Algorithm 1*, the neighbour selection procedure is shown. Based on preliminarily analysed movie datasets, we create predictions for any active user utilising the extracted time-bin size in Fig. 1 (a) and Fig. 1 (b). The active user rating date for a test item used as the starting point. By going back in the timeline, previous ratings are considered, and N item ratings are captured. This is the indication of user activation frequency to mitigate the temporal side-effects. This process is performed in line 2 of *Algorithm 1*. Then, variable-sized prospective neighbour time-bins are extracted in line 5. There are two underlying constraints: (i) having the test item rated and (ii) having other commonly rated items concerning the test user time-bin. Neighbours can have flexible time-bin size and all possible time-bins are extracted by obeying $5 \geq N_{prospective\ neighbor} \geq 100$. If there are less than 5 items, an exception occurs based on our requirements, and if there are more than 100 items, performance bottleneck is observed.

Between lines 8 and 18 of *Algorithm 1*, prospective neighbour time-bins are selected; accordingly, PCCs are computed beforehand concerning the test user. PCC has a higher priority in neighbourhood calculation. In line 10, *sorting* is applied first to observe the dominant correlation values between user time-bins. After that, any prospective neighbour is processed for the commonly rated items in the time-bin of interest. Among various neighbour time-bin possibilities, only that with the higher number of common elements and higher PCC value is selected. Finally, in line 19 of *Algorithm 1*, k neighbours are elected with the highest privileged time-bins based on PCC. After *Algorithm 1*, Eq. (2) is called for test item prediction. The performance of the proposed approach is numerically measured in the next section.

4 EXPERIMENTAL RESULTS

In this section, we evaluate the prediction performance of the proposed approach. The traditional method with standard neighbour selection and the time-bin-based approach were tested comparatively using the performance metrics given in Section 2. During the evaluations, multi-class classification via confusion matrix construct was utilised, as mentioned. We examined the results at the rating granularity, addressing the Netflix and MovieLens datasets. Thus, more detailed analyses at the rating levels are presented with rating-wise monitoring.

The test phase contains two different perspectives. The first is that the static N value is forced for the test user. A test person is expected to watch and rate at least N movies in the time-bin, which may cause a cold start problem [21]. Therefore, in the second phase of experiments, we removed this requirement by flexibly allowing test users who watch $< N$ movies. This means that if any test user has watched less than N movies in the timeline before the test item, this exact amount in the $[5,\ N)$ interval can construct the time-bin. The lower limit is still 5, similar to that in prospective neighbour time-bins. In Figs. 3-6, the performance metrics for the first experimental phase are illustrated by setting N as constant. If a user has less than N movie ratings, this user is skipped both for the time-bin-based method and for the standard approach without time-bin. Hence, the same *'N film constraint'* is applied for both approaches, thereby making a fair comparison between the two. In the figures below, bar plots represent *with* and *without* the time-bin-based approaches. In dark coloured bars, the standard approach does not consider time-bin-based correlation calculations; instead, it considers the whole user history.

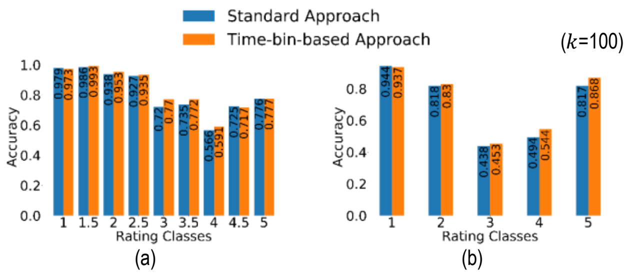


Figure 3 Accuracy (with static N), (a) MovieLens and (b) Netflix

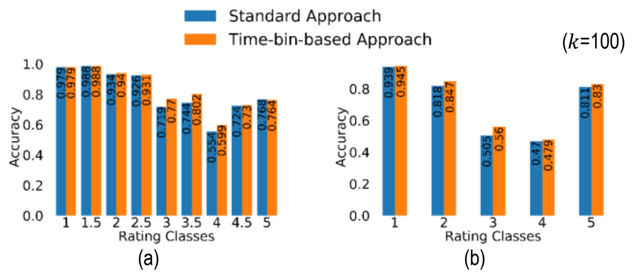


Figure 7 Accuracy (with flexible N), (a) MovieLens and (b) Netflix

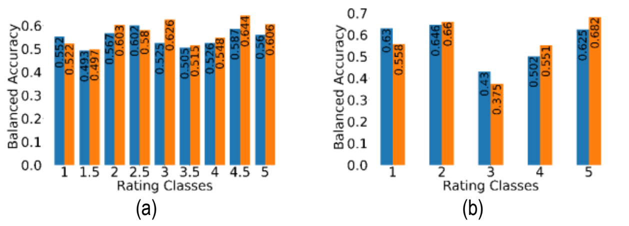


Figure 4 Balanced accuracy (with static N), (a) MovieLens and (b) Netflix

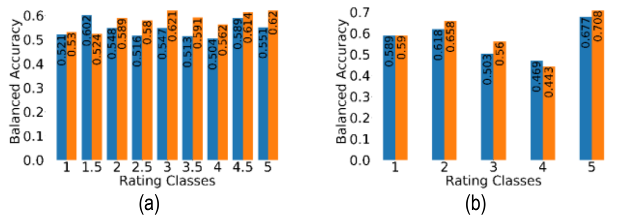


Figure 8 Balanced accuracy (with flexible N), (a) MovieLens and (b) Netflix

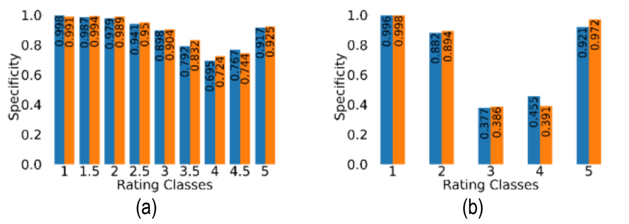


Figure 5 Specificity (with static N), (a) MovieLens and (b) Netflix

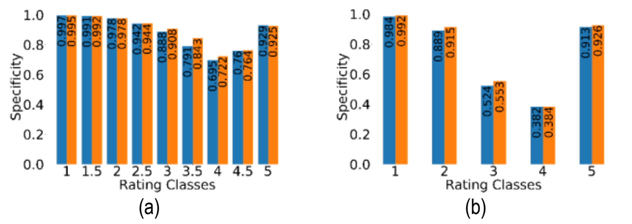


Figure 9 Specificity (with flexible N), (a) MovieLens and (b) Netflix

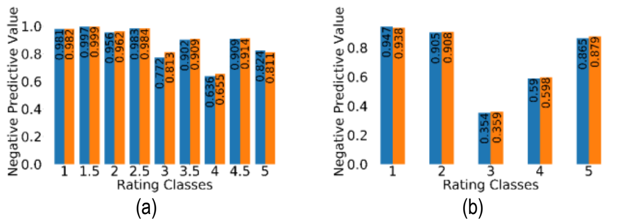


Figure 6 NPV (with static N), (a) MovieLens and (b) Netflix

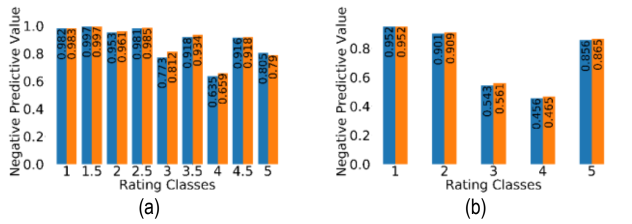


Figure 10 NPV (with flexible N), (a) MovieLens and (b) Netflix

According to rating-based inspection, each rating class is presented with the corresponding performance metrics. We employed the accuracy and balanced accuracy metrics to monitor the overall performance. In addition, specificity and inverse precision, i.e., NPV, were considered for the negative classification performance. We randomly selected users from MovieLens and Netflix Prize for the test. The results presented here denote the average of 18 000 distinct test attempts. From the above figures, it can be observed that when rating-based results are averaged into a single score, the time-bin-based approach outperforms the values in the standard method for all metrics.

We also unveiled the cold start problem of static N, and the test user time bin was allowed to be flexible down to 5. To this end, the same metrics are monitored for both datasets as presented in Figs. 7-10. Similarly, the performance of the proposed method is valid; additionally, the metric scores slightly increase, on average, if the test user time-bin is flexible.

Owing to rating-wise observations, a notable finding from the results is the user judgement on the highest/lowest rating values. This inference shows that the maximum *likes* and *dislikes* have relatively higher scores on the metrics, which is crucial for binary prediction. Furthermore, the performance of ‘3’ and ‘4’ rating values is relatively lower. These rating classes source essential information for the threshold of user judgments that are also utilised for the performance metrics. The applied algorithm alters the *threshold* in the Netflix dataset when active user time-bin size flexibility is allowed. From Figs. 3-6 (b) to Figs. 7-10 (b), the threshold changes from ‘3’ to ‘4’ for the Netflix dataset.

As a general remark, negative predictions made in the tests presented higher ratios than positive predictions. On observing the balanced accuracy and specificity, it can be concluded that the TP rate is less in comparison to the TN rate. The standard approach obeys the N film restrictions for fairness; thus, the overall gain for both approaches demonstrates that the TN predictions are more effective.

Finally, other than the bar plots above, we separately observed a compound metric, MCC, that yielded an average score from the rating values. MCC presents a wide implicit observation, especially for binary predictions. The score improvements were recorded for both datasets for the time-bin-based approach. However, the best performance was observed in the MovieLens dataset, where MCC was in the range [0,3 , 0,4], similar to the Netflix dataset. Compared to the standard approach, the MCC improved by +7,44% and +5,36% for MovieLens and Netflix datasets, respectively, when *Algorithm 1* was applied with flexible N.

5 CONCLUSIONS

In this study, we presented an approach to boost the prediction performance of an RS framework using temporal features. The time-bin-based method enhanced the top-k neighbour selection procedure and the prediction accuracy was improved. After analysing the datasets preliminarily for global temporal information, the time-bin size was determined for the test user. PCC was utilised for user-user similarity weights, in which the rating history of the test user was restricted to the last N items. Neighbours, whose time-bin size can be flexible, were selected using the calculated weights and thereby implicitly the common rating counts. The algorithm was tested using the *Netflix Prize* and *MovieLens* datasets. The possibility of cold start problem can be reduced if the time-bin size of the test user is flexible. Improvements were observed in all performance metrics for the proposed approach. In future work, the algorithm can be extended to other temporal user actions, such as *date of system registration* and *total time spent online on an application*.

Acknowledgements

xxxxx xxxxx.

6 REFERENCES

[1] Schafer, J. Ben, Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.) *The Adaptive Web: Methods and Strategies of Web Personalization*, Springer Berlin Heidelberg, 291-324. https://doi.org/10.1007/978-3-540-72079-9_9

[2] Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261-273. https://doi.org/10.1016/j.eij.2015.06.005

[3] Guo, S., Zhang, W., & Zhang, S. (2017). A pagerank-based collaborative filtering recommendation approach in digital libraries. *Tehnički Vjesnik*, *24*(4), 1051-1058. https://doi.org/10.17559/TV-20160602011232

[4] Sharma, M., Ahuja, L., & Kumar, V. (2020). A novel rule based data mining approach towards movie recommender system. *Journal of Information and Organizational Sciences*, *44*(1), 157-170. https://doi.org/10.31341/jios.44.1.7

[5] Odić, A., Tkalčič, M., Tasič, J. F., & Košir, A. (2013). Impact of the context relevancy on ratings prediction in a movie-recommender system. *Automatika*, *54*(2), 252-262. https://doi.org/10.7305/automatika.54-2.258

[6] Kapoor, S., Gupta, V., & Kumar, R. (2018). An obfuscated attack detection approach for collaborative recommender systems. *Journal of Computing and Inf. Technology*, *26*(1), 45–56. https://doi.org/10.20532/cit.2018. 1003948

[7] Harper, F. M., & Konstan, J. A. (2015). The MovieLens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems*, *5*(4), 1-19. https://doi.org/10.1145/2827872

[8] Grouplens. (1992). *MovieLens*. Retrieved from https://grouplens.org/datasets/movielens/

[9] Netflix. (2009). *Netflix Prize*. Retrieved from https://www.netflixprize.com/

[10] Xue, Y. (2021). A dynamic trust relations-based friend recommendation algorithm in social network systems. *Tehnički Vjesnik*, *28*(1), 185-192. https://doi.org/10.17559/TV-20200825171016

[11] Koenigstein, N., Dror, G., & Koren, Y. (2011). Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. in *Proceedings of the Fifth ACM Conference on Recommender Systems*, 165-172. https://doi.org/10.1145/2043932.2043964

[12] Koren, Y. (1993). Collaborative filtering with temporal dynamics. *Proceedings of the 15th ACM SIGKDD Inter. Conference on Knowledge Discovery and Data Mining*, 447–456, https://doi.org/10.1145/1557019.1557072

[13] Bakir, C. (2018). Collaborative filtering with temporal dynamics with using singular value decomposition. *Tehnički Vjesnik*, *25*(1), 130–135. https://doi.org/10.17559/TV-20160708140839

[14] Li, L., Zheng, L., Yang, F., & Li, T. (2014). Modeling and broadening temporal user interest in personalized news recommendation. *Expert Systems with Applications*, *41*(7), 3168-3177. https://doi.org/10.1016/j.eswa.2013.11.020

[15] Lathia, N., Hailes, S., Capra, L., & Amatriain, X. (2010). Temporal diversity in recommender systems. *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 210-217. https://doi.org/10.1145/1835449.1835486

[16] Al-Hadi, I. A. A., Sharef, N. B. M., Sulaiman, N., & Mustapha, N. (2017). Review of the temporal recommendation system with matrix factorization. *International Journal of Innovative Computing, Information and Control*, *13*(5), 1579-1594. Retrieved from http://www.ijicic.org/ijicic-130511.pdf

[17] Sivaramakrishnan, N., *et al*. (2018). Neighborhood-based approach of collaborative filtering techniques for book recommendation system. *International Journal of Pure and Applied Mathematics*, *119*(12), 13241-13250. Retrieved from http://eprints.rclis.org/33266/1/1207.pdf

[18] Liu, H., Hu, Z., Mian, A., Tian, H., & Zhu, X. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems*, vol. *56*, 156-166. https://doi.org/10.1016/j.knosys.2013.11.006

[19] Zhang, Z., Kudo, Y., & Murai, T. (2017). Neighbor selection for user-based collaborative filtering using covering-based rough sets. *Annals of Operations Research*, *256*, 359-374. https://doi.org/10.1007/s10479-016-2367-1

[20] Rokach, L. (2009). Taxonomy for characterizing ensemble methods in classification tasks: A review and annotated bibliography. *Computational Statistics and Data Analysis*, *53*(12), 4046-4072. https://doi.org/10.1016/j.csda.2009.07.017

[21] Nadimi-Shahraki, M. H., & Bahadorpour, M. (2014). Cold-start problem in collaborative recommender systems: Efficient methods based on ask-to-rate technique. *Journal of Computing and Information Technology*, *22*(2), 105–113. https://doi.org/10.2498/cit.1002223

Contact information:

XXXX XXXX XXXX XXXX XXXX
xxxxxx xxxxxx xxxxxx xxxxxx
xxxxxx xxxxxx xxxxxx xxxxxx
xxxxxx xxxxxx xxxxxx xxxxxx