

Speeding Up Recommender Systems Using Association Rules [★]

Eyad Kannout^{1,2}[0000–0001–7543–774X]
 Hung Son Nguyen^{1,3}[0000–0002–3236–5456]
 Marek Grzegorowski^{1,4}[0000–0003–4740–0725]

¹ Institute of Informatics, University of Warsaw, Warsaw, Poland

² eyad.kannout@mimuw.edu.pl

³ son@mimuw.edu.pl

⁴ m.grzegorowski@mimuw.edu.pl

Abstract. Recommender systems are considered one of the most rapidly growing branches of Artificial Intelligence. The demand for finding more efficient techniques to generate recommendations becomes urgent. However, many recommendations become useless if there is a delay in generating and showing them to the user. Therefore, we focus on improving the speed of recommendation systems without impacting the accuracy. In this paper, we suggest a novel recommender system based on Factorization Machines and Association Rules (FMAR). We introduce an approach to generate association rules using two algorithms: (i) apriori and (ii) frequent pattern (FP) growth. These association rules will be utilized to reduce the number of items passed to the factorization machines recommendation model. We show that FMAR has significantly decreased the number of new items that the recommender system has to predict and hence, decreased the required time for generating the recommendations. On the other hand, while building the FMAR tool, we concentrate on making a balance between prediction time and accuracy of generated recommendations to ensure that the accuracy is not significantly impacted compared to the accuracy of using factorization machines without association rules.

Keywords: Recommendation system · Association rules · Apriori algorithm · Frequent pattern growth algorithm · Factorization machines · Prediction's time · Quality of recommendations.

1 Introduction

Throughout ¹ the past decade, recommender systems have become an essential feature in our digital world due to their great help in guiding the users towards the most likely items they might like. Recently, recommendation systems

[★] Research co-funded by Polish National Science Centre (NCN) grant no. 2018/31/N/ST6/00610.

¹ The final publication is available at Springer via https://doi.org/10.1007/978-3-031-21967-2_14

have taken more and more place in our lives, especially during the COVID-19 pandemic, where many people all over the world switched to online services to reduce the direct interaction between each other. Many researchers do not expect life to return to normal even after the epidemic. All previous factors made recommender systems inevitable in our daily online journeys.

Many online services are trying to boost their sales by implementing recommendation systems that estimate users' preferences or ratings to generate personalized offers and thus recommend items that are interesting for the users. Recommendation systems can be built using different techniques which leverage the rating history and possibly some other information, such as users' demographics and items' characteristics. The goal is to generate more relevant recommendations. However, these recommendations might become useless if the recommendation engine does not produce them in a proper time frame.

Recently, the factorization machine has become a prevalent technique in the context of recommender systems due to its capabilities of handling large, sparse datasets with many categorical features. Although many studies have proved that factorization machines can produce accurate predictions, we believe that the prediction time should also be considered while evaluating this technique. Therefore, in this paper, we work on finding a novel approach that incorporates association rules in generating the recommendations using the factorization machines algorithm to improve the efficiency of recommendation systems. It is worth noting that the factorization machine model is used to evaluate our method and compare the latency of FM before and after using the association rules. However, in practice, our method can be combined with any other recommendation engine to speed up its recommendations.

The main contributions of this paper are as follows: 1) proposing a method that uses the apriori algorithm or frequent pattern growth (FP-growth) algorithm to generate association rules which suggest items for every user based on the rating history of all users; 2) utilizing these association rules to create short-listed set of items that we need to generate predictions for them; 3) employing factorization machines model to predict missing user preferences for the short-listed set of items and evaluate the top-N produced predictions.

The remainder of this paper is organized as follows. In Section 2, we provide background information for factorization machines algorithm and association rules in addition to reviewing some related works. In Section 3, we describe the problem we study in this paper. Also, we present FMAR - a novel recommender system that utilizes factorization machines and association rules to estimate users' ratings for new items. Section 4 evaluates and compares FMAR with a traditional recommender system built without employing association rules. Finally, in Section 5, we conclude the study and suggest possible future work.

2 Preliminaries

In this section, we briefly summarize the academic knowledge of factorization machines and association rules.

2.1 Factorization Machines

In linear models, the effect of one feature depends on its value. While in polynomial models, the effect of one feature depends on the value of the other features. Factorization machines [1] can be seen as an extension of a linear model which efficiently incorporates information about features interactions, or it can be considered equivalent to polynomial regression models where the interactions among features are taken into account by replacing the model parameters with factorized interaction parameters. [2].

However, polynomial regression is prone to overfitting due to a large number of parameters in the model. Needless to say, it is computationally inefficient to compute weights for each interaction since the number of pairwise interactions scales quadratically with the number of features. On the other hand, factorization machines elegantly handled previous issues by finding a one-dimensional vector of size k for each feature. Then, the weight values of any combination of two features can be represented by the inner product of the corresponding features vectors. Therefore, factorization machines manage to factorize the interactions weight matrix $W \in \mathbb{R}^{n \times n}$, which is used in polynomial regression, as a product VV^T , where $V \in \mathbb{R}^{n \times k}$. So, instead of modeling all interactions between pairs of features by independent parameters like in polynomial regression (see Equation 1). We can achieve that using factorized interaction parameters, also known as latent vectors, in factorization machines (see Equation 2).

$$\hat{y}(x) = w_0 + \sum_{i=1}^N w_i x_i + \sum_{i=1}^N \sum_{j=i+1}^N w_{ij} x_i x_j \quad (1)$$

$w_0 \in \mathbb{R}$: is the global bias.

$w_i \in \mathbb{R}^n$: models the strength of the i -th variable.

$w_{ij} \in \mathbb{R}^{n \times n}$: models the interaction between the i -th and j -th variable.

$$\hat{y}(x) = w_0 + \sum_{i=1}^N w_i x_i + \sum_{i=1}^N \sum_{j=i+1}^N \langle v_i, v_j \rangle x_i x_j \quad (2)$$

$\langle v_i, v_j \rangle$: models the interaction between the i -th and j -th variable by factorizing it, where $V \in \mathbb{R}^{n \times k}$ and $\langle \cdot, \cdot \rangle$ is the dot product of two vectors of size k .

This advantage is very useful in recommendation systems since the datasets are mostly sparse, and this will adversely affect the ability to learn the feature interactions matrix as it depends on the feature interactions being explicitly recorded in the available dataset.

2.2 Association Rules

The basic idea of association rules [3][4] is to uncover all relationships between elements from massive databases. These relationships between the items are extracted using every distinct transaction. In other words, association rules try to

find global or shared preferences across all users rather than finding an individual's preference like in collaborative filtering-based recommender systems.

At a basic level, association rule mining [3][4][5] analyzes data for patterns or co-occurrences using machine learning models. An association rule consists of an antecedent, which is an item found within the data, and a consequent, which is an item found in combination with the antecedent. Various metrics, such as support, confidence, and lift, identify the most important relationships and calculate their strength. Support metric [3][4][5] is the measure that gives an idea of how frequent an itemset is in all transactions (see Equation 3). The itemset here includes all items in antecedent and consequent. On the other hand, the confidence [3][4][5] indicates how often the rule is true. In other words, it defines the percentage of occurrence of consequent given that the antecedents occur (see Equation 4). Finally, the lift [5] is used to discover and exclude the weak rules that have high confidence, which can be calculated by dividing the confidence by the unconditional probability of the consequent (see Equation 5). Various algorithms are in place to create associations rules using previous metrics, such as Apriori [4][6], AprioriTID [4][6], Apriori Hybrid [4][6], AIS (Artificial Immune System) [7], SETM [8] and FP-growth (Frequent pattern) [4][9]. In the next section, we provide more details about how we use these metrics to find the association rules used to improve the prediction time in the recommender system.

$$\text{Support}(\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both X and Y}}{\text{Total Number of transactions}} \quad (3)$$

$$\text{Confidence}(\{X\} \rightarrow \{Y\}) = \frac{\text{Transactions containing both X and Y}}{\text{Transactions containing X}} \quad (4)$$

$$\text{Lift}(\{X\} \rightarrow \{Y\}) = \frac{\text{Confidence}}{\text{Transactions containing Y}} \quad (5)$$

2.3 Related works

Over the past decade, a lot of algorithms concerned with improving the accuracy of the recommendation have been constantly proposed. However, while reviewing the research literature related to recommendation systems and what has been done to improve the prediction's time, we find that there is a research gap in this area even though the speed of recommendation, besides the accuracy, is a major factor in real-time recommender systems.

Xiao et al. [10] worked on increasing the speed of recommendation engines. They spotted that the dimension of the item vector in a collaborative filtering algorithm is usually very large when we calculate the similarity between two items. To solve this problem, they introduced some methods to create a set of expert users by selecting small parts of user data. One of these methods is based on selecting expert users according to the number of types of products they have purchased before. In comparison, another method calculates the similarities between users and then selects expert users based on the frequency that the user appears in other users' K-most similar users set. The results show that using expert users in an item-based collaborative filtering algorithm has increased

the speed of generating recommendations with preserving the accuracy to be very close to original results. Tapucu et al. [11] carried out some experiments to check the performance of user-based, item-based, and combined user/item-based collaborative filtering algorithms. Different aspects have been considered in their comparisons, such as size and sparsity of datasets, execution time, and k-neighborhood values. They concluded that the scalability and efficiency of collaborative filtering algorithms need to be improved. The existing algorithms can deal with thousands of users within a reasonable time. Still, modern e-commerce systems require to scale to millions of users and hence, expect even improved prediction time and throughput of recommendation engines. According to previous findings, we believe there is room for further improvements concerning prediction time and efficiency of recommendation systems.

3 FMAR Recommender System

In this section, we formally provide the statement of the problem that we aim to tackle. Then, we introduce the details of a novel recommender system that is based on Factorization Machine and Association Rules (FMAR). We first formalize the problem we plan to solve. Then, we describe our proposed model which has two versions based on the algorithm used to generate the association rules: (i) factorization machine apriori based model, and (ii) factorization machine FP-growth based model.

3.1 Problem Definition

In many recommender systems, the elapsed time required to generate the recommendations is very crucial. Moreover, in some systems, any delay in generating the recommendations can be considered as a failure in the recommendation engine. The main problem we address in this paper is to minimize the prediction latency of the recommender system by incorporating the association rules in the process of creating the recommender system. The main idea is to use the association rules to decrease the number of items that we need to approximate their ratings, hence, decreasing the time that the recommender system requires to generate the recommendations. Also, our goal is to make sure that the accuracy of the final recommendations is not impacted after filtering the items using the association rules.

3.2 Factorization Machine Apriori Based Model

In this section, we introduce the reader to the first version of FMAR which proposes a hybrid model that utilizes factorization machines [1] and apriori [4][6] algorithms to speed up the process of generating the recommendations. Firstly, we use apriori algorithm to create a set of association rules based on the rating history of users. Secondly, we use these rules to create users' profile which

recommends a set of items for every user. Then, when we need to generate recommendations for a user, we find all products that are not rated before by this user, and instead of generating predictions for all of them, we filter them using the items in the users' profile. Finally, we pass the short-listed set of items to a recommender system to estimate their ratings using factorization machines model.

In the context of association rules, it is worth noting that while generating the rules, all unique transactions from all users are studied as one group. On the other hand, while calculating the similarity matrix in collaborative filtering algorithms, we need to iterate over all users and every time we need to identify the similarity matrix using transactions corresponding to a specific user. However, what we need to do to improve the recommendation speed is to generate predictions for parts of items instead of doing that for all of them. Next, we introduce the algorithms that we use to generate the association rules and users' profile (cf. Algorithm 1).

Algorithm 1 Association Rules Generation Using Apriori Algorithm

```

1: Extract favorable reviews                                ▷ ratings > 3
2: Find frequent item-sets  $\mathcal{F}$                         ▷ support > min_support
3: Extract all possible association rules  $\mathcal{R}$ 
4: for each  $r \in \mathcal{R}$  do
5:   Compute confidence and lift
6:   if (confidence < min_confidence) or (lift < min_lift) then
7:     Filter out this rule from  $\mathcal{R}$ 
8:   end if
9: end for
10: Create users' profile using rules in  $\mathcal{R}$ 

```

Algorithm 2 Users' Profile Generation

```

1: for each user do
2:   Find high rated items based on rating history
3:   Find the rules that their antecedents are subset of high rated items
4:   Recommend all consequences of these rules
5: end for

```

After generating the users' profile, we can use it to improve the recommendation speed for any user by generating predictions for a subset of not-rated items instead of doing that for all of them. The filtering is simply done using the recommended items which are extracted for every user using the association rules (cf. Algorithm 2). Moreover, the filtering criteria can be enhanced by using the recommended items of the closest n-neighbors of the target user. The similarity between the users can be calculated using pearson correlation or cosine similarity measures.

On the other hand, it is noteworthy that the association rules in our experiments are generated using the entire dataset which means that these rules try to find global or shared preferences across all users. However, another way to generate the association rules is to split the dataset based on users' demographics, such as gender, or items' characteristics, such as genre, or even contextual information, such as weather and season. Thus, if we are producing recommendations for a female user in winter season, we can use dedicated rules which are extracted from historical ratings given by females in winter season. Following this strategy, we can generate multiple sets of recommendation rules which can be used later during prediction time to filter the items. Obviously, the rules generated after splitting the dataset will be smaller. So, the prediction latency can be minimized by selecting a smaller set of rules. In fact, this feature is very useful when we want to make a trade-off between the speed and quality of recommendations. Lastly, it is important to note that several experiments are conducted in order to select the appropriate values of hyper-parameters used in previous algorithms. For instance, $\text{min_support} = 250$, $\text{min_confidence} = 0.65$, number of epochs in FM = 100, number of factors in FM = 8. However, multiple factors are taken into consideration while selecting those values, including accuracy, number of generated rules, and memory consumption.

Algorithm 3 FP-Tree Construction

```

1: Find the frequency of 1-itemset
2: Create the root of the tree (represented by null)
3: for each transaction do
4:   Remove the items below min.support threshold
5:   Sort the items in frequency support descending order
6:   for each item do ▷ starting from highest frequency
7:     if item not exists in the branch then
8:       Create new node with count 1
9:     else
10:      Share the same node and increment the count
11:    end if
12:  end for
13: end for

```

3.3 Factorization Machine FP-Growth based Model

In this section, we introduce the second version of FMAR where FP-growth [4] [9] algorithm has been employed to generate the association rules. In general, FP-growth algorithm is considered as an improved version of apriori method which has two major shortcomings: (i) candidate generation of itemsets which could be extremely large, and (ii) computing support for all candidate itemsets which is computationally inefficient since it requires scanning the database many times. However, what makes FP-growth algorithm different from apriori algorithm is the fact that in FP-growth no candidate generation is required. This is achieved

by using FP-tree (frequent pattern tree) data structure which stores all data in a concise and compact way. Moreover, once the FP-tree is constructed, we can directly use a recursive divide-and-conquer approach to efficiently mine the frequent itemsets without any need to scan the database over and over again. Next, we introduce the steps followed to mine the frequent itemsets using FP-growth algorithm. We will divide the algorithm into two stages: (i) FP-tree construction, and (ii) mine frequent itemsets (cf. Algorithm 3 and Algorithm 4).

Algorithm 4 Mining Frequent Itemsets

- 1: Sort 1-itemset in frequency support ascending order
 - 2: Remove the items below min_support threshold
 - 3: **for each** 1-itemset **do** ▷ starting from lowest frequency
 - 4: Find conditional pattern base by traversing the paths in FP-tree
 - 5: Construct conditional FP-tree from conditional pattern base
 - 6: Generate frequent itemsets from conditional FP-tree
 - 7: **end for**
-

After finding the frequent itemsets, we generate the association rules and users' profiles in the same way as in FM Apriori-based model. Regarding the hyper-parameters of FP-Growth algorithm, we used min_support = 60 and min_confidence = 0.65. Finally, in order to generate predictions, we employ a factorization machines model, which is created using the publicly available software tool libFM [12]. This library provides an implementation for factorization machines in addition to proposing three learning algorithms: stochastic gradient descent (SGD) [1], alternating least-squares (ALS) [13], and Markov chain Monte Carlo (MCMC) inference [14].

4 Evaluation for FMAR

In this section, we conduct comprehensive experiments to evaluate the performance of the FMAR recommender system. In our experiments, we used MovieLens 100K dataset¹ which was collected by the GroupLens research project at the University of Minnesota. MovieLens 100K is a stable benchmark dataset that consists of 1682 movies and 943 users who provide 100,000 ratings on a scale of 1 to 5. It is important to note that, in this paper, we are not concerned about users' demographics and contextual information since the association rules are generated based only on rating history.

4.1 Performance Comparison and Analysis

In order to provide a fair comparison, we use several metrics and methods to evaluate FMAR and FM recommender systems, such as Mean Absolute Error

¹ <https://grouplens.org/datasets/movielens/>

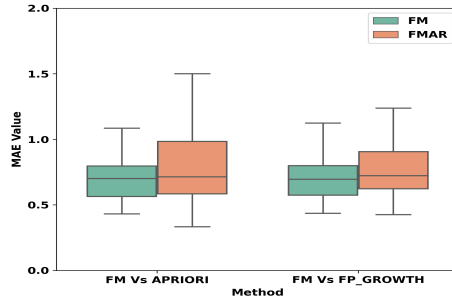


Fig. 1: MAE Comparison

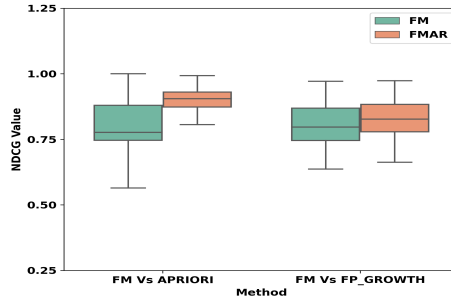


Fig. 2: NDCG Comparison

(MAE), Normalized Discounted Cumulative Gain (NDCG), and Wilcoxon Rank-Sum test. Firstly, we selected 50 users who made a significant amount of ratings in the past. For every user, a dedicated testing set has been created by arbitrary selecting 70% of the ratings made by this user in the past. On the other hand, the training set is constructed using the rest of the records in the entire dataset, which are not used in testing sets. This training set is used to generate the association rules and build the factorization machines model. For each evaluation method, we created two sets of items for every user. The first one, called *original*, contains all items in the testing set. While the second one, called *short-listed*, is created by filtering the original set using the association rules. Finally, we pass both sets to the factorization machines model to generate predictions and evaluate both versions of FMAR, i.e., Apriori-based and FP-growth-based FM models, by comparing them with the standard FM model operating on the complete data.

In the first experiment, we calculate the mean absolute error (MAE) generated in both recommendation engines. The main goal of this approach is to show that the quality of recommendations is not significantly impacted after filtering the items in the testing set using the association rules. Fig 1 compares the mean absolute error of the predictions made using FM model with FM Apriori model and FM FP-growth model for 50 users. We use a box plot, which is a standardized way of displaying the distribution of data, to plot how the values of mean absolute error are spread out for 50 users. This graph encodes five characteristics of the distribution of data which show their position and length. These characteristics are minimum, first quartile (Q1), median, third quartile (Q3), and maximum. The results of this experiment show that MAE of FMAR in both versions, FM Apriori and FM FP-growth, is very close to MAE of FM recommender system. However, the average value of MAE for 50 users is 0.71 using FM model, 0.80 using FMAR (Apriori model), and 0.78 using FMAR (FP Growth model).

In the second experiment, we evaluate FMAR by comparing its recommendation with FM using Normalized Discounted Cumulative Gain (NDCG) which is a measure of ranking quality that is often used to measure the effectiveness

of recommendation systems or web search engines. NDCG is based on the assumption that highly relevant recommendations are more useful when appearing earlier in the recommendations list. So, the main idea of NDCG is to penalize highly relevant recommendations that appear lower in the recommendation list by reducing the graded relevance value logarithmically proportional to the position of the result. Fig 2 compares the accuracy of FM model with FM Apriori model and FM FP-growth model for 50 users and shows the distribution of the results. It is worth noting that in this test we calculate NDCG using the highest 10 scores in the ranking which are generated by FM or FMAR. The results show that both versions of FMAR model have always higher NDCG values than FM model which means that true labels are ranked higher by predicted values in FMAR model than in FM model for the top 10 scores in the ranking.

In the third evaluation method, we run Wilcoxon Rank-Sum test on the results of previous experiments. Firstly, we apply Wilcoxon Rank-Sum test to the results of the first experiment in order to prove that the difference in MAE between FM and FMAR is not significant, and hence, it can be discarded. So, we pass two sets of samples of MAE for FM and FMAR. The Table 1 shows the p-value for comparing FMAR using Apriori model and FP Growth model with FM model. In both cases, we got p-value > 0.05 which means that the null hypothesis is accepted at the 5% significance level, and hence, the difference between the two sets of measurements is not significant. On the other hand, we apply Wilcoxon Rank-Sum test to the results of the second experiment to check if the difference in NDCG between FM and FMAR is significant. However, the Table 1 shows that p-value < 0.05 for comparing both models of FMAR with FM model. This means the null hypothesis is rejected at the 5% significance level (accept the alternative hypothesis), and hence, the difference is significant. Since FMAR model has higher NDCG than FM model, we can conclude that FMAR outperforms FM for the highest top 10 predictions.

Model	p-value
MAE (FM Vs FMAR-Apriori model)	0.29
MAE (FM Vs FMAR-FP Growth model)	0.13
NDCG (FM Vs FMAR-Apriori model)	1.74e-08
NDCG (FM Vs FMAR-FP Growth model)	0.04

Table 1: Wilcoxon Rank-Sum Test

In the last experiment, we compare FM and FMAR in terms of the speed of their operation, measured as the number of predictions performed by the factorization machines model. The main idea is to estimate the time necessary to prepare recommendations for every tested user for both evaluated approaches.

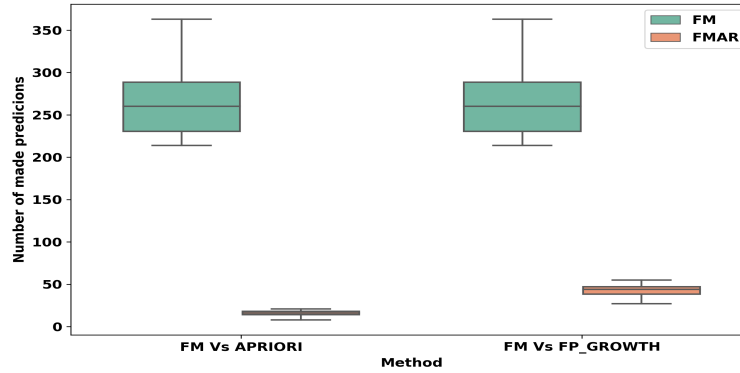


Fig. 3: Comparison of the speed of methods (estimated by the number of predictions made by the factorization machine model, i.e., the lower the better)

Fig 3 shows the distribution of the results of this experiment for the selected 50 test users. Observably, the number of items that we need to predict with FMAR is significantly lower due to using the association rules for filtering. The results show that the FMAR model can generate predictions for any user at least four times faster than the FM model. Finally, it is noteworthy that generating the rules is part of the training procedure. Therefore, it is a one-time effort, and there is no need to regenerate or update the association rules frequently in FMAR. Therefore, the computational cost of the training procedure for every method, including extracting the association rules, is not considered in our comparisons.

In the final analysis, all previous experimentations showed that after applying our method, the factorization machines could perform significantly faster with no drop in quality considering MAE and NDCG measures.

5 Conclusions and Future Work

This article introduces FMAR, a novel recommender system, which methodically incorporates the association rules in generating the recommendations using the factorization machines model. Our study evaluates two approaches to creating association rules based on the users' rating history, namely: the apriori and frequent pattern growth algorithms. These rules are used to decrease the number of items passed to the model to estimate the ratings, reducing the latency of the recommender system prediction.

To evaluate our proposed model, we conducted comprehensive experiments on MovieLens 100K dataset using the libFM tool [12] which provides implementations for factorization machines as well as machine learning algorithms. Moreover, we presented multiple evaluation methods to compare the performance of FMAR against the recommender system built using the factorization machines algorithm. The experimental results show that FMAR has improved

the efficiency of recommender systems. Furthermore, the experiments also indicate that the accuracy of FMAR is very close to the results produced by the standard recommender system.

In the future work, we plan to incorporate more information in the process of producing the association rules, such as users' demographics, items' characteristics, and contextual information. Another important aspect to consider is to evaluate our proposed model using different recommender systems and different sizes of datasets. However, we are interested in creating a web interface where FMAR is used to generate recommendations for users. In this scenario, we are particularly interested in employing more advanced algorithms to generate the association rules, such as AprioriTID [4][6], Apriori Hybrid [4][6], AIS (Artificial Immune System) [7] and SETM [8]. We believe that using previous algorithms would help to further improve the performance and accuracy of FMAR. As a result, FMAR can be evaluated using different settings selected in a user-friendly web interface.

Furthermore, we plan to consider the changes in users' behavior and preferences by periodically updating association rules based on recent changes in rating history. Another direction of future work is to utilize the generated association rules to solve the cold-start problem in recommendation systems where the new users do not have (or have very few) ratings in the past. Finally, we plan to use distributed stream processing engines, like Apache Flink, to examine parallel implementations of FMAR, where the process of extracting the rules and generating the recommendations is scalable to vast streams or large-scale datasets.

References

1. S. Rendle, "Factorization Machines", in Proc. IEEE Int. Conf. Data Mining, Dec. 2010, pp. 995–1000, doi: 10.1109/ICDM.2010.127.
2. C. Freudenthaler, L. Schmidt-Thieme and S. Rendle, "Factorization Machines Factorized Polynomial Regression Models", 2011.
3. W. Haotong, "Data Association Rules Mining Method Based on Improved Apriori Algorithm", In 2020 the 4th International Conference on Big Data Research (ICBDR'20) (ICBDR 2020). Association for Computing Machinery, New York, NY, USA, 12–17. <https://doi.org/10.1145/3445945.3445948>.
4. N.Satyavathi, B.Rama and A.Nagaraju, "Present State-of-The-Art of Dynamic Association Rule Mining Algorithms", International Journal of Engineering and Advanced Technology, Volume-9 Issue-1, October 2019, PP: 6398-6405, doi: 10.35940/ijeat.A2202.109119.
5. B. Fuguang, L. Mao, Y. Zhu, C. Xiao, and C. Xu, "An Improved Evaluation Methodology for Mining Association Rules". Axioms 2022, Volume:11, ISSN:2075-1680, doi: 10.3390/axioms11010017.
6. Merry Kp, Rabindra Kumar Singh and Swaroop. S. Kumar, "Apriori-hybrid algorithm as a tool for colon cancer microarray data classification," International Journal of Engineering Research and Development, vol.4, pp. 53-57, 2012.
7. K. Khurana and S. Sharm, "A comparative analysis of association rule mining algorithms", Journal of Scientific and Research Publications, Volume 3, Issue 5, 2013.

8. A. Saxena and V. Rajpoot, "A Comparative Analysis of Association Rule Mining Algorithms", IOP Conference Series: Materials Science and Engineering, Volume 1099, 2021, doi: 10.1088/1757-899X/1099/1/012032.
9. Y. Zeng, S. Yin, J. Liu, and M. Zhang, "Research of Improved FP-Growth Algorithm in Association Rules Mining", Scientific Programming Journal, 2015, ISSN:1058-9244, doi: 10.1155/2015/910281.
10. W. Xiao, S. Yao, and S. Wu, "Improving on recommend speed of recommender systems by using expert users", Chinese Control and Decision Conference (CCDC), 2016: 2425–30, doi: 10.1109/CCDC.2016.7531392.
11. D. Tapucu, S. Kasap, and F. Tekbacak, "Performance comparison of combined collaborative filtering algorithms for recommender systems" in 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops, July 2012, pp. 284–289, doi: 10.1109/COMPSACW.2012.59.
12. S. Rendle, "Factorization Machines with libFM" ACM Trans. Intell. Syst. Technol. 3, 3, Article 57 (May 2012), 22 pages, doi: 10.1145/2168752.2168771.
13. S. Rendle, Z. Gantner, C. Fredenthale, and L. Schmidit-Thieme, "Fast context-aware recommendations with factorization machines", In Proceedings of the 34th ACM SIGIR Conference on Research and Development in Information Retrieval, 2011, doi: 10.1145/2009916.2010002.
14. C. Fredenthaler, L. Schmidit-Thieme, and S. Rendle, "Bayesian factorization machines" In Proceedings of the NIPS Workshop on Sparse Representation and Low-rank Approximation, 2010.