IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 21 December 2023, accepted 7 January 2024, date of publication 17 January 2024, date of current version 30 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3355057

RESEARCH ARTICLE

Clustering-Based Frequent Pattern Mining Framework for Solving Cold-Start Problem in Recommender Systems

EYAD KANNOUT, MAREK GRZEGOROWSKI, MICHAŁ GRODZKI, AND HUNG SON NGUYEN
Institute of Informatics, University of Warsaw, 02-097 Warsaw, Poland

Corresponding author: Eyad Kannout (eyad.kannout@mimuw.edu.pl)

This work was supported by the Polish National Science Centre (NCN) under Grant 2018/31/N/ST6/00610.

ABSTRACT Recommender systems (RS) are substantial for online shopping or digital content services. However, due to some data characteristics or insufficient historical data, may encounter considerable difficulties impacting the quality of their recommendations. This study introduces the clustering-based frequent pattern mining framework for recommender systems (Clustering-based FPRS) - a novel RS constituting several recommendation strategies leveraging agglomerative clustering and FP-growth algorithms. The developed strategies combine the generated frequent itemsets with collaborative- and content-filtering methods to address the cold-start problem, which occurs whenever a new user or item enters the system. In such cases, the RS has limited information about the new user or object. Thus, the recommendations may be inaccurate. The experimental evaluation on several benchmark datasets showed that Clustering-based FPRS is superior to state-of-the-art and could effectively alleviate the cold-start problem.

INDEX TERMS Cold-start problem, recommender system, frequent pattern mining, clustering.

I. INTRODUCTION

Modern businesses reach outstanding amounts of customers, who are often overwhelmed by the plethora of products and services [1], [2]. Online stores and content services attempt to provide a personalized shopping experience, tailoring their offerings to the specific interests of each client [1], [2], [3]. Viable solutions to this challenge are recommender systems (RS), which leverage the data collected to provide users with the most appealing products or services they are looking for [4]. The fundamental idea of RS is to recommend items that are of interest to a user based on their previous interactions with comparable ones or categories, ratings' history of users with similar preferences, and possibly some other information, such as users' demographics or items' characteristics [5].

Recommender systems are typically classified into three main categories: (i) collaborative filtering (CF), (ii) content-based filtering (CBF), and (iii) hybrid methods. The basic idea behind collaborative filtering is that users with much the same taste or preferences tend to behave in a cognate way in the future. Those techniques rely on historical interactions to compute similarities among users for whom the recommendations are eventually generated. An analogous approach can be applied to create recommendations based on item similarities. On the other hand, content-based filtering tries to utilize items' characteristics, users' demographics, and contextual information to recommend additional items comparable to those preferred by the target user in the past. Finally, hybrid techniques cover the weaknesses and exploit the strengths of CF and CBF models by combining them to provide more relevant results [6].

However, due to some data characteristics, RS may encounter significant difficulties, and the quality of their recommendations is strictly limited to the density of interactions in the collected data, which is often very sparse [7]. Furthermore, they may be severely affected by the dynamic changes in the commercial environment where

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Fiumara.

producers contend to astonish clients with ever newer and better offers. Dealing with still-emerging products and services, as well as with new visitors, is one of the most challenging tasks faced by state-of-the-art recommender systems, often referred to as the cold start problem [5], [8]. This phenomenon is particularly inconvenient and occurs whenever recommendations are generated for a new item without a proper history of user interactions or ratings. Similarly, user cold start refers to the situation when a recommendation engine faces a new visitor, and there is no historical data about their personal preferences. In fact, many state-of-the-art recommendation algorithms may generate unreliable recommendations for such cases since they cannot learn the preference embedding of these new users/items [9], [10].

In this study, we introduce Clustering-based FPRS, that significantly extends the FPRS recommender system [11], [12] by providing several strategies for devising more effective and efficient recommendations that successfully support new (cold) items and users. The developed strategies are organized into user cold-start and item cold-start modules framing a comprehensive recommendation platform for data scientists and analysts to select the optimal solution for the investigated problem and data characteristics. Clustering-based FPRS employs available contextual information to alleviate the cold-start problem more effectively and relies on the FP-growth algorithm to generate frequent patterns based on users' and items' characteristics.

Clustering-based FPRS resolves the problem of data sparsity by operating on the more granular representation of similar users and items instead of operating on the very sparse user-item matrix. We derive this intermediate data representation by grouping similar users and items together. For that purpose, we rely on distance-based clustering to reflect and leverage content-based similarities between items and users, which is material for partially incomplete data. By incorporating agglomerative techniques, Clustering-based FPRS can tune the number (and size) of clusters and search for the optimal thresholds. It also gives the possibility to utilize multiple-value characteristics to measure the similarity. The dense matrix has a positive impact on the FP-growth algorithm and the generated frequent itemsets.

The main contributions of this paper are as follows:

1) We introduce *Clustering-based Frequent Pattern mining framework for Recommender Systems* constituting of several recommendation strategies to mitigate the cold-start problem
2) We utilize the FP-growth algorithm to produce frequent itemsets based on the ratings in the user-item matrix.
3) We mitigate the sparsity issue by converting the user-item rating matrix into a lower-dimensional one with clustering techniques.
4) We incorporate contextual information and extend the recommendation list using similarity measures of users and items to address the cold-start problem in the FPRS framework.
5) We conducted comprehensive experiments of the proposed approach on several benchmark datasets, including MovieLens 100K, MovieLens 1M, HetRec-MovieLens, and LDOS-CoMoDa, and evaluated Clustering-based FPRS against several state-of-the-art recommender systems.

The remainder of this paper is organized as follows. Section II presents research efforts on the cold-start problem in the domain of recommender systems. In Section III, we provide background information related to recommender systems (in Section III-A), frequent pattern mining (Section III-B), and hierarchical clustering (Section III-C). In Section IV, we present the clustering-based frequent pattern mining framework for recommender systems (Clustering-based FPRS), a novel hybrid RS that copes with the sparsity issue as well as the cold-start problem. In Section IV-A, we discuss multiple strategies to address the cold-start for new users and new items. Section V evaluates the proposed model against several baseline models, which are designed to mitigate the cold-start problem in recommender systems. Finally, in Section VI, we summarise the study and discuss possible future research directions.

II. RELATED WORKS

Recommender systems (RS) are a rapidly growing field of computer science and intelligent systems that allow users to find items of interest that best meet their preferences, and the cold-start problem is one of the biggest challenges we face while designing RS. Specifically, it concerns the issue that the recommender system cannot draw any inferences for new users, who signed up recently to the system, or new items, which added recently to the system, since it has not yet gathered sufficient information to generate reliable recommendations. Therefore, the RS (and especially the collaborative methods) suffers from both user cold-start and item cold-start problems due to deficient information about new entities [13], [14].

Most of the attempts to address such a problem suggest combining collaborative filtering methods with content-based approaches that utilize the intrinsic characteristics of the analyzed entities. For example, in [9], the authors propose a hybrid RS that combines the content-based filtering and latent Dirichlet allocation (LDA)-based models to address the cold-start problem. In [15], we may find a hybrid RS that combines a recommender module composed of a collaborative filtering system (using the singular value decomposition algorithm), a content-based system, and a fuzzy expert system. On the other hand, the content-based filtering (CBF) methods are capable to recommend new items even if there are no ratings provided by users. However, these methods suffer from the user cold-start problem due to the insufficient number of rated items [6], [16].

In literature, many efforts have been made to cope with the cold-start problem [17], [18]. In [19], the authors utilized the concept of weak supervision to address the item-side cold-start problem by combining content-based filtering and the preferences of representative users. In [20], the authors extended the matrix-factorization-based methods, namely SVD, SVD++, and the NMF models, to address the cold-start problem. The basic idea is to use three simple regularization differentiating functions (RDF) so that the regularization weights on different items and users are set based on their popularity. However, we used this model as a baseline with which to compare the performance of our proposed method (cf. Section V-B). Another interesting approach explores the ability of collective matrix factorization models in recommender systems (CMFREC) to make predictions about users and items for which there is side information available but no feedback or historical ratings/interactions data [21]. The cold-start recommendations generated by this approach were found to be of better quality than non-personalized recommendations, and predictions about new users were found to be more reliable than those about new items. This model is also used as a benchmark to evaluate the performance of our proposed model (cf. Section V-B)

There are many more attempts to alleviate the cold-start problem by combining collaborative filtering with content-based methods, including using simultaneous co-clustering [22], meta-learning [23], or Siamese neural networks [10]. Moreover, many efforts have been made to employ various dimensionality reduction techniques in the field of recommender systems [24]. Considering the discussed problem of missing or insufficient information, it seems interesting to refer to the dimensionality reduction methods based on the granularization of the attribute space [25], and particularly on resilient ML techniques [26], [27] - i.e., resistant to data deficiencies. The hybridization of soft computing techniques with collaborative and content-based methods is a wide-ranging field of research, and an interesting area for the further development of recommendation systems [28], particularly interesting for context-aware RSs [29], [30], [31].

Another approach to mitigating cold-start is to recommend the recent trend in users' preferences, or in other words, to return the most recent popular items [32], [33]. However, the recommendations made by this approach are not always reliable and result in so-called popularity bias since users often differ in their preferences, which may also vary between types of products and their characteristics [34]. Hence, an additional effort to deal with biases in data is required [35]. Another interesting approach to dealing with insufficient or missing historical transactions was to exploit additional sources of information. In particular, in [36], the authors enhance the data representation about new entities by training RSs with the Linked Open Data model based on DBpedia. Another way is to directly inquire the users about their preferences. Such information may be collected, e.g., via survey or by asking users to select the most relevant picture related to the desired item [37].

On the other hand, addressing the cold-start problem by combining community-based knowledge with association rule mining is also showing very promising results [14]. Referring to association rule mining (cf. [38]) and frequent pattern mining (cf. [39]) techniques to cope with the cold-start problem is useful also when it comes to minimizing the latency of recommender systems [40]. For this reason, frequent pattern mining is particularly interesting in our research, and we provide a detailed overview of the literature on this field in Section III-B.

While reviewing the literature, we spotted that most reported cases focus on alleviating cold users [17], [18] rather than investigating the scenarios related to new items. Hence, we believe that such approaches deserve further research attention having in mind the still-emerging new products and services alongside the high demand for finding more efficient techniques to generate recommendations. Additionally, we did not find in the literature any attempt to address the cold-start problem using frequent pattern mining methods. Furthermore, the proposed method in this study can address the sparsity nature of the user-item rating matrix by employing the agglomerative clustering technique which produces granular representations of this matrix. The new dense user-item matrix gives an additional improvement for frequent itemsets generated by the FP-growth algorithm. Moreover, our method can employ contextual information in the clustering step to generate more relevant recommendations. The contextual information can be either combined with users' and items' characteristics or even used individually in case of missing other features. All previous aspects make our study novel and potentially significant, considering the efficiency of our method in terms of accuracy and computational time. The cold-start problem is still one of the most prevailing topics deserving further attention and is particularly interesting in the context of our study [10], [36].

III. BACKGROUND KNOWLEDGE

In this section, we briefly summarize the academic knowledge of recommender systems, frequent pattern mining, and clustering techniques.

A. OVERVIEW OF RECOMMENDER SYSTEMS

Recommender Systems (RSs) are a collection of algorithms and techniques that aim to provide personalized recommendations for items to a particular user. In this section, we provide an overview of recommender systems and discuss the most popular approaches used in this area, shedding some light on how these techniques suffer from the cold-start problem.

To mathematically formulate the problem of a recommender system, let us assume there is a list of users $U = \{u_1, u_2, \ldots, u_n\}$ and a list of items $I = \{i_1, i_2, \ldots, i_m\}$. Let R be the user-item rating matrix, where R_{ui} denotes the rating given by user u to item i. The task is to predict the missing ratings in R and provide personalized recommendations to users based on their preferences. Formally, this can be expressed as finding a function that estimates the unknown ratings (see Equation 1). The goal is to create a recommendation model that minimizes the difference between the predicted ratings $f(u, i)$ and the actual ratings R_{ui} for observed user-item pairs, and generalizes well to unseen user-item pairs and provides accurate predictions or rankings for items a user may like.

$$f : U \times I \rightarrow \mathbb{R} \tag{1}$$

Collaborative filtering (CF) is one of the most widely used and successful techniques in RS, with a wide range of application prospects in many fields, such as e-commerce and social networks [41], [42]. CF techniques do not need any knowledge about intrinsic users' or items' characteristics. Instead, they utilize historical interactions between them [43]. The simple motivation behind this assumes that certain groups of users share tastes and tend to behave similarly in the future, and the best implicit way to establish this information is to compare their former choices [11].

CF-based methods operate on the collected interactions between users (e.g., customers in the webshop, book readers, or viewers of short videos in Internet services) and items (e.g., webpages, videos, or books) to generate recommendations. The interactions may have the form of transactions or ratings. The more ratings the users provide, the more precise the outcomes, generated with higher confidence.

Recommender systems rely on various types of input since the ratings or preferences can be acquired explicitly or implicitly [43]. In the first case, users rate items, hence expressing their preferences. The latter refers to the situation when the ratings are inferred from observable user activity, such as page views or mouse clicks, etc. [30]. Furthermore, collaborative filtering methods can be divided into two main categories - namely, *memory-based* and *model-based* techniques.

The memory-based RS relies on the stored user-items interactions, typically represented as a matrix. This class of algorithm employs various notions of similarity between vectorized representations of users (or items), which may be computed, for instance, by referring to distance metrics (e.g., Euclidean, Manhattan, or Chebyshev distance). Hence, the memory-based methods are further divided into user-based and item-based.

The user-based methods search for other users in the dataset whose interests are close to the investigated one in terms of their former ratings' similarity [44]. Using their top-rated products to predict what the target user would like is a common and efficient heuristic. On the other hand, the item-based collaborative filtering is an analogous procedure to the above-described one. Here, for every product that the target user has not rated before, we estimate the rating using the closest neighbors which are rated previously by the target user. It is important to note that the degree of similarity is considered in the process of rating estimation. However, the selection of a proper similarity measure is considered a key decision. A plethora of possible functions can be used, such as cosine similarity or Pearson correlation coefficient.

The model-based CF algorithms opt to fit a predictive model to further use this model for predicting new ratings [45]. The model-building process may be implemented based on machine learning algorithms, such as matrix factorization (MF) [46], [47], Bayesian classifiers [48], or neural networks and fuzzy systems [49], [50]. In recent years, it was also suggested to use a multilayer perceptron, which is often referred to as neural collaborative filtering. Still, methods based on matrix factorization are most broadly used due to their ability to address the efficient way they handle scalability and sparsity issues [51], [52]. One of the most popular techniques applied to solve the matrix factorization problem in this regard is singular value decomposition (SVD) [53].

Although CF-based methods are very efficient and give good results, they still suffer from the cold-start problem which occurs whenever a recommender system tries to generate recommendations for either a new user who signed up recently to the system without having any rating records available yet or when a new item is added to the system without any rating given to that item so far. In fact, most state-of-the-art recommendation algorithms generate unreliable recommendations for such cases since they cannot learn the preference embedding of these new users/items [13], [54].

Content-based Filtering (CBF) has become a relevant approach in the development of recommender systems. In contrast to collaborative filtering, content-based filtering does not depend on rating co-occurrences across the users. Instead, it utilizes the characteristics of an item to recommend items similar to those a given user has liked in the past [55]. It is based on the fact that items with similar attributes will be rated similarly. While CBF is very useful, it requires a profound knowledge of each item, in addition to a user profile describing the user's taste. So, the task is to learn the user preferences and then use the most similar items to the ones already evaluated positively by a user to generate a list of recommendations. [56]. The recommendation process in content-based filtering is performed in three steps [33]: (i) item profiling, (ii) user profiling, and (iii) recommendation generation [33]. Item profiling, or item representation, is the process of extracting a set of features from a dedicated item. In practice, the item representation can be divided into two main categories: (i) structured and (ii) unstructured. In the structured representation, each item is represented by a set of attributes that have limited and specific values, such as director, actors, and genres in the case of movies. On the other hand, in the unstructured representation, such as articles content, the attribute value is unlimited and requires conversion into structured data, such as vectors, on which we can compute metrics, such as Euclidean distance, Pearson's coefficient, or cosine similarity [57].

However, content-based filtering methods are capable to recommend new items which have been added recently to the system and most likely do not have, or have very few, ratings in the past. Conversely, the recommendations generated by collaborative filtering techniques are negatively impacted in this scenario since they solely rely on the historical ratings [6], [54], [58]. However, in content-based recommender systems, it is necessary to learn user preferences in order to provide reliable recommendations. Therefore, CBF suffers from the user cold-start problem when new users who signed up recently do not have, or have very few, ratings, and hence, the quality of recommendation will be impacted by an insufficient number of rated items [6], [16].

The hybrid filtering model is a combination of more than one filtering approach. It is introduced to overcome some common problems that are associated with filtering approaches, such as the cold-start problem, over-specialization problem, and sparsity problem [59]. For example, the sparsity problem is addressed in the hybrid recommendation model by integrating the information of the content-based filtering and collaborative filtering models. Another motivation behind the implementation of hybrid filtering is to improve the accuracy and efficiency of the recommendation process. However, the outcomes of any hybrid model strongly depend on the used algorithms and the method of hybridization, for example, how the outcomes of an algorithm relate to the other ones [6]. Basically, hybrid methods can be implemented in various ways as follows [58]:

- Implement collaborative and content-based methods individually, then aggregate their predictions to yield better recommendations.
- Integrate some content-based characteristics into a collaborative method. This greatly helps to address the cold-start problem in collaborative filtering and the over-specialization problem of content-based filtering.
- Integrate some collaborative characteristics into a content-based method.
- Construct a single unified recommendation model that integrates both content-based and collaborative characteristics to improve the effectiveness of the recommendation process.

B. FREQUENT PATTERN MINING

The main objective of frequent pattern mining, also known as association rule mining, is to discover useful patterns and relationships between elements in large databases using every distinct transaction. In contrast to collaborative filtering which aims to find the individual preferences for each user, frequent pattern mining seeks to discover global or shared preferences across all users. [60].

Each association rule consists of an antecedent and a consequent, both of which are a list of items. However, three metrics, support, confidence, and lift, are in place to analyze the dataset, identify the most important frequent patterns (itemsets), and finally extract the association rules. Support [61] is the measure that gives an idea of how frequently an itemset appears in all transactions. In other words, the support is measured by the proportion of transactions in which an itemset appears. While confidence [61] indicates how likely the rule is true. It defines the percentage of transactions containing the consequent given that the antecedent exists. The last metric, called lift, is considered as a correlation measure to find and exclude the weak rules that have high confidence. The Equations 2, 3, and 4 show how we calculate support, confidence, and lift, respectively.

$$support(A \Rightarrow B) = P(A \cup B) \tag{2}$$

$$confidence(A \Rightarrow B) = P(B|A) = \frac{support(A \cup B)}{support(A)} \tag{3}$$

$$lift(A \Rightarrow B) = \frac{P(A \cup B)}{P(A)P(B)} = \frac{support(A \cup B)}{support(A)support(B)} \tag{4}$$

Various algorithms exist for mining frequent itemsets, such as Apriori [62], [63], AprioriTID [62], [63], Apriori Hybrid [62], [63], Eclat (Equivalence CLAss Transformation) [64], [65], and FP-growth (Frequent pattern) [62], [66]. In this paper, we utilize FP-growth algorithm to generate frequent itemsets since it has several advantages over other algorithms. For instance, Apriori algorithm suffers from two major shortcomings: (i) the large size of candidate itemsets and (ii) the high costs of disk I/O and computing power due to multiple scans of the database. All these problems are solved in FP-growth algorithm by leveraging the FP-tree (frequent pattern tree) data structure to store all data in a concise and compact way. Therefore, the size of candidate itemsets will not be a problem anymore. 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 like in other algorithms [66]. In this way, the cost of searching frequent patterns is substantially reduced. Finally, it is worth mentioning that extensive research efforts have been made by practitioners in order to find the major issues/challenges related to algorithms used for frequent pattern mining in addition to considering the performance efficiency of these algorithms by applying them to real-world datasets under varying conditions [67], [68]. The comparative analysis shows that FP-growth algorithm is the most efficient/scalable among all other algorithms. Therefore, we decided to use FP-growth algorithm in our proposed framework (Clustering-based FPRS) to generate frequent itemsets.

C. RUDIMENTS OF CLUSTERING TECHNIQUES

In this section, we provide a brief introduction to clustering techniques focusing on hierarchical methods which are used in our experiments to cluster the users and items.

Data clustering is an unsupervised learning technique designed for creating groups of objects (clusters) so that objects within the same cluster are very similar and objects in different clusters are quite distinct. Mathematically, assuming there is a dataset with n objects, each of which is described by m features, is denoted by $D = \{x_1, x_2, \ldots, x_n\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{im})^T$ is a vector denoting the ith object and x_{ij} is a scalar denoting the jth component or feature of x_i. The number of features m is also called the dimensionality of the dataset [69]. The clustering algorithm is the process of assigning a class label $l_i \in \{1, 2, \ldots, k\}$ to each object x_i to identify its cluster class, where k is the number of clusters. Mathematically, clustering for a given dataset can be represented by an assignment function $f : D \rightarrow [0, 1]^k, x \rightarrow f(x)$, defined as follows:

$$f(x) = (f_1(x), f_2(x), \ldots, f_k(x)), \tag{5}$$

$$\sum_{i=1}^{k} f_i(x) = 1, \quad \forall x \in D, \tag{6}$$

As a result, f is representing hard clustering if for every $x \in D, f_i(x) \in \{0, 1\}$, otherwise, it is representing soft (fuzzy) clustering [69]. In other words, in hard clustering, each object should belong to one and only one cluster, while in fuzzy clustering, the object can belong to more than one cluster with probabilities. Next, we investigate the hierarchical clustering techniques since they are employed in our proposed framework.

Clustering techniques are referred to as hierarchical if the number of clusters changes when the algorithm proceeds. Hence, there is no need to determine the number of clusters at the beginning of the algorithm. Hierarchical algorithms are classified into two types: divisive hierarchical algorithms and agglomerative hierarchical algorithms [70].

In an agglomerative hierarchical algorithm, the algorithm proceeds from the bottom to the top. The algorithm starts from n clusters with every single object in a single cluster. Then, it recursively merges two disjoint clusters according to some distance measure to form a new cluster until all of the data are in a single cluster. It is worth noting that there are several agglomerative hierarchical methods, such as single-link method and complete link method, which can be distinguished according to different distance measures they use [70], [71].

On the other hand, the divisive hierarchical algorithm can be viewed as a top-bottom clustering method. Initially, the algorithm starts with putting all objects in one large cluster, and then recursively splits it into smaller clusters according to some similarity criteria. DIANA algorithm [70] is one of the most popular divisive hierarchical algorithms. At each step in DIANA algorithm, the cluster with the largest diameter is split until each object is in a single cluster at step $n - 1$. The diameter of cluster C is defined to be the largest dissimilarity between two objects.

IV. CLUSTERING-BASED FREQUENT PATTERN MINING FRAMEWORK

The main problem we address in this paper is to mitigate the impact of new users and items on collaborative filtering techniques. This section introduces the clustering-based frequent pattern mining framework for recommender systems (Clustering-based FPRS), a hybrid RS that significantly mitigates the cold-start problem.

Figure 1 depicts the process of generating the recommendations consisting of five stages: (i) Data Input, (ii) Data Preparation, (iii) Data Preprocessing, (iv) Frequent Pattern Mining, and (v) Recommendation Generation. In the first stage, the user-item rating matrix is enriched by selected content-based users' and items' characteristics. In the data preparation stage, we provide data quality analysis, filtering, and verification of selected features' importance. During the data preprocessing stage, the user-item rating matrix is converted into a lower-dimensional one using clustering techniques. Following this transformation, the matrix is partitioned based on selected features, resulting in the creation of smaller matrices. In the fourth stage, we generate frequent itemsets using the FP-growth algorithm. Finally, we produce the recommendations with the FPRS framework that consists of two main modules: user cold-start and item cold-start module. Instead of providing one method that fits all purposes, we follow multiple strategies in our framework. Therefore, each module offers several approaches to selecting the features and producing recommendations (cf. Section IV-A).

The main idea behind the Clustering-based FPRS is to convert the user-item rating matrix into a lower-dimensional one using clustering techniques. Mathematically, let us assume there is a dataset with n users and η features. The result of clustering algorithms can be represented by a $k \times n$ matrix as follows:

$$U = \begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1k} \\ u_{21} & u_{22} & \ldots & u_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \ldots & u_{nk} \end{pmatrix}, \tag{7}$$

where u_{ij} denotes the membership of user i in cluster j, and k denotes the number of clusters. This approach allows us to generalize collaborative filtering techniques to operate on clusters of similar users. The most typical case considers that each user belongs to only one cluster meeting the condition:

$$u_{ij} \in \{0, 1\}, \quad 1 \leq i \leq n, \; 1 \leq j \leq k \tag{8}$$

However, by application of fuzzy clustering, we can model more complex scenarios in which users may (to some extent, e.g., expressed by probabilities) belong to one or more clusters:

$$u_{ij} \in [0, 1], \quad 1 \leq i \leq n, 1 \leq j \leq k \tag{9}$$

$$\sum_{j=1}^{k} u_{ij} = 1, \quad 1 \leq i \leq n \tag{10}$$

$$\sum_{i=1}^{n} u_{ij} > 0, \quad 1 \leq j \leq k. \tag{11}$$

An analogous approach is applied to items' representation. The consequence of applying the clustering techniques on users and items is a clustered user-item rating matrix with a significantly higher rating density (cf. the upper right part of Figure 1). Depending on some selected criteria related to data characteristics or quality revealed in the *"Attribute analysis"* phase of the process on Figure 1, such as feature distribution or missing values' ratio, we select the most appropriate features to build users' and items' representation.

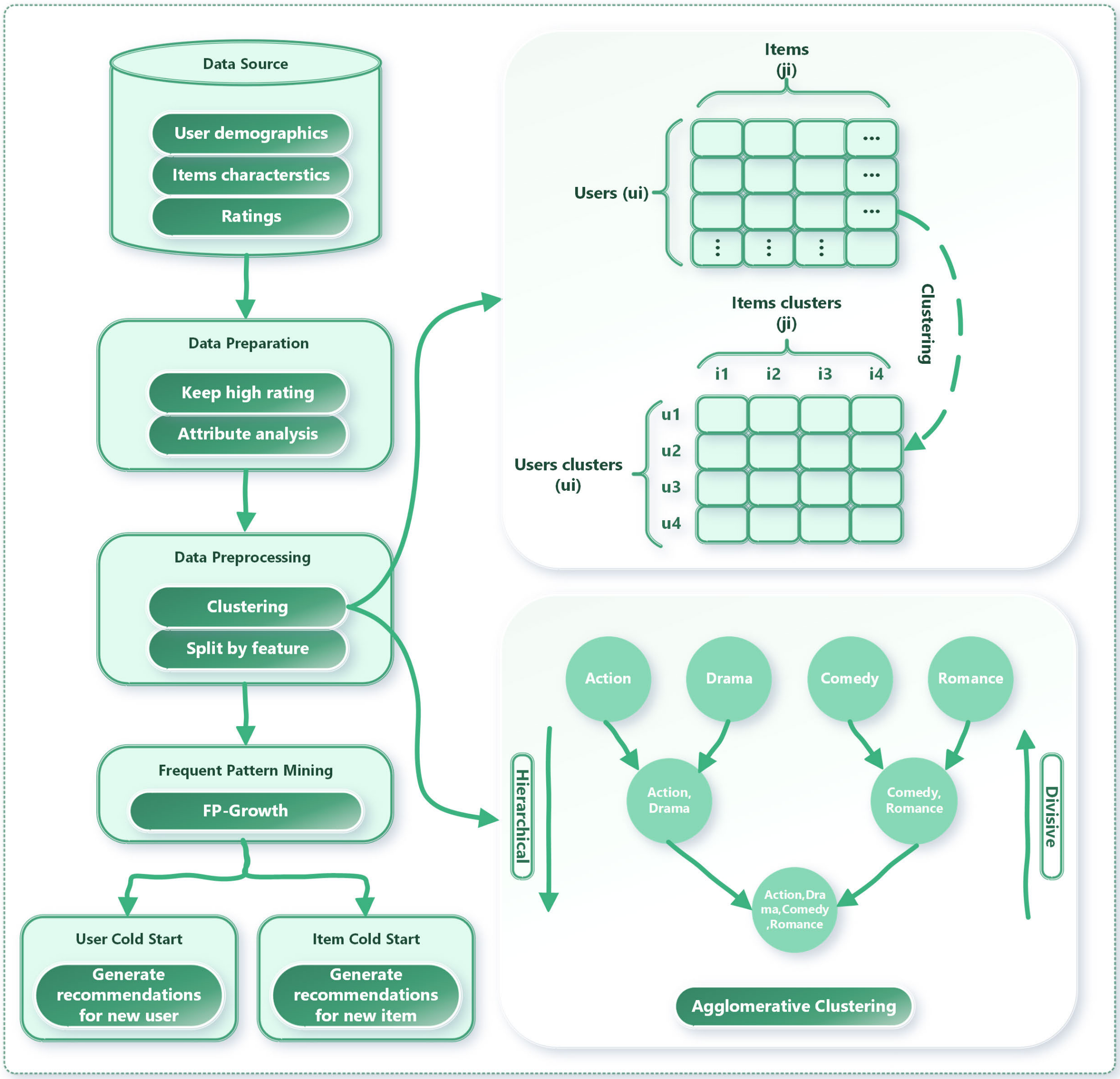


FIGURE 1. **Clustering-based frequent pattern mining framework for recommender systems.**

Moreover, the contextual information can be utilized as well to cluster the users and items [30].

It is well known that using datasets with high rating density gives the means to generate more frequent itemsets, leading to a better quality of recommendations. The introduced clustered user-item matrix is instrumental in mitigating the cold-start problem. We include it in all recommendation algorithms introduced in Section IV-A. As shown in the experiments in Section V, by incorporating clustering, the FPRS strategies of the user and item cold-start modules could significantly alleviate the cold-start problem related to both new users and items.

In the presented study, we implement hierarchical clustering (cf. Figure 1). There are, however, many more ways we can extend this approach in the future. One of the approaches, which could respond to the over-specialization of recommendations (i.e., the serendipity problem), is to apply fuzzy logic. By referring to fuzzy clustering, we may define the probability that a particular user belongs to each cluster. These probabilities can be considered as weights to estimate the prediction by computing the dot product of two vectors - the user's probabilities and the aggregated rating for that item's cluster per each user's cluster. More formally:

$$s(i,j) = a_i \cdot b_j = \sum_{r=1}^{k} a_{ir} b_{jr} \tag{12}$$

where $s(i,j)$ represents the predicted rating of user i for item cluster j, a_i is the vector expressing the probabilities/weights of user i within each user's cluster, and b_j is the vector representing the aggregated ratings (e.g., average, median, or a range of min-max values) of item cluster j for each user's cluster. The length of both vectors should be equal and represents the number of user clusters k (cf. Figure 1). Finally, we can select some items from the clusters with the highest estimated rating and recommend them to the target user. We can analogously apply this process to recommend a dedicated item to a list of users.

The cold-start problem can be categorized into two types. The complete cold-start problem, where no interactions are present for new users, and the incomplete cold-start problem, where there are only a few interactions for new users. It is essential to emphasize that the Clustering-based FPRS is designed to handle both incomplete and complete cold-start scenarios. The resolution for the incomplete cold-start challenge involves transforming the user-item matrix into a lower dimension, resulting in a clustered matrix that is less sparse than the original one (cf. stage Data Preprocessing in Figure 1). By operating on the clustered matrix, classical collaborative filtering techniques can generate more accurate recommendations, particularly for users or items with limited ratings (referred to as warm users or items). On the other hand, all the strategies outlined in Section IV-A contribute to resolving the complete cold-start problem. By employing these strategies, it becomes possible to generate recommendations for users or items lacking any historical ratings (referred to as cold users or items).

A. USER AND ITEM COLD-START MODULES

In the user cold-start module of our framework, we focus on generating recommendations for new users who signed up recently to the system and have a very small rating history. To generate recommendations in such a situation, we use several Strategies 1, 2, 3, and 4, which mainly differ in three factors: (i) features selected to split the data, (ii) the type of generated frequent patterns (itemsets or usersets) and (iii) the way how the frequent patterns are utilized to generate the recommendations.

In Strategy 1, we do not split the dataset, but rather we extract frequent 1-itemsets for the entire dataset and recommend these popular items to the new users. In Strategy 2, we split the data by users' demographics, and then for each user group, created by splitting the data, we use the items in the generated frequent 1-itemsets to compose the recommendation set that will eventually be recommended to new users based on their demographics (the user groups they belong to). On the other hand, in Strategies 3 and 4, we generate frequent usersets instead of itemsets. Here, every frequent userset contains the users that highly rated the same movies more than the *min_support* threshold. In both Strategies 3 and 4, the items, rated by frequent usersets and fulfilled the participation threshold, are utilized to form the recommendation set for a dedicated user group. Figure 2 shows how we can use the gender feature, which can be replaced or extended by any other users' features, in Strategy 3 to generate recommendations for the cold users.

Strategy 1 User Cold-Start Module (Popularity)
```
Data:
  dataTrain - entire training dataset (no records splitting)
  Res - recommendation list for new user
  $\rho$ - list of frequent 1-itemsets
  $\theta$ - acceptable support threshold
Procedure:
  $\rho \leftarrow$ fp-growth(dataTrain, $\theta$)
  $Res \leftarrow \rho$ (only frequent 1-itemsets)
Recommendation:
  Recommend all items in Res to the new_user
```

However, in Strategy 4, we split the data using users' and items' characteristics, so the recommendation set is created for each user group using the frequent usersets that exist in the group containing the largest number of these frequent sets (called the dominant group).

The item cold-start module aims to generate recommendations for new items with a minimal rating record. We align with the position that providing one ultimate solution to address the cold-start problem in all datasets may be unobtainable. Therefore, we follow multiple strategies to generate such recommendations. More details about the developed approaches are provided in Strategies 5, 6, and 7.

In Strategy 5, we split the data only by items' characteristics, while in Strategy 6, we utilize both items' and users' characteristics to split the data. Moreover, for each item group in Strategies 5 and 6, we compose the recommendation set by selecting the users who are involved in generating the frequent itemsets more than the participation threshold. Then, we recommend the new item to a set of users based on the item group it belongs to. However, the users in the recommendation set are selected from all frequent itemsets in Strategy 5.

In Strategy 6, the users are selected from the user group that contains the largest number of frequent itemsets, called the dominant group, since in this strategy we split the data by users' and items' characteristics. On the other hand, in Strategy 7, we form clusters based on the frequent 1-itemsets created by users' and items' features. Then, the recommendation set for the new item is produced based on the percentage of users' engagement in creating the frequent itemsets in the closest cluster.

Now, let us provide a detailed analysis of the time (computational) complexity for all strategies used in item and user modules. Initially, the time complexity of the FP-growth algorithm that identifies all frequent itemsets of size k is $O(n^2)$ [72], where k is the number of elements in the itemset. However, the complexity decreases when we impose constraints on the size of frequent itemsets we are seeking. For instance, when $k = 1$, the complexity of FP-growth becomes linear. In all formulas describing the complexity of strategies, n_u denotes the total number of users, whereas n_i denotes the total number of items.

Strategy 2 User Cold-Start Module (Popularity in User Groups)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - recommendation list for new user
$\vartheta$ - selected users features and contextual information
$\rho$ - frequent 1-itemsets per userGroup
$\theta$ - acceptable support threshold
$\omega$ - items in frequent_itemsets
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho$ $\leftarrow$ fp-growth(dataTrainSets, $\theta$)
for each frequent 1-itemsets in $\rho$(userGroup) do
    $Res$(userGroup) $\leftarrow$ $\omega$
end for
Recommendation:
$\gamma$ $\leftarrow$ findUserGroup(new_user)
Recommend all items in $Res(\gamma)$ to the new_user
```

Strategy 3 User Cold-Start Module (User Groups)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - recommendation list for new user
$\vartheta$ - selected users features and contextual information
$\rho$ - frequent usersets per userGroup
$\theta$ - acceptable support threshold
$\delta$ - acceptable participation percentage threshold
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho$ $\leftarrow$ fp-growth(dataTrainSets, $\theta$)
for each frequent_usersets in $\rho$(userGroup) do
    for each item do
        if item_participation $> \delta$ then
            $Res$(userGroup) $\leftarrow$ item
        end if
    end for
end for
Recommendation:
$\gamma$ $\leftarrow$ findUserGroup(new_user)
Recommend all items in $Res(\gamma)$ to the new_user
```

Strategy 4 User Cold-Start Module (Dominant Groups)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - recommendation list for new user
$\vartheta$ - selected users and items features and contextual information
$\rho$ - frequent usersets per userItemGroup
$\theta$ - acceptable support threshold
$\delta$ - acceptable participation percentage threshold
$\beta$ - dominant frequent usersets per userGroup (largest usersets)
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho$ $\leftarrow$ fp-growth(dataTrainSets, $\theta$)
$\beta$ $\leftarrow$ findDominantGroups($\rho$)
for each frequent_usersets in $\beta$(userGroup) do
    for each item do
        if item_participation $> \delta$ then
            $Res$(userGroup) $\leftarrow$ item
        end if
    end for
end for
Recommendation:
$\gamma$ $\leftarrow$ findUserGroup(new_user)
Recommend all items in $Res(\gamma)$ to the new_user
```

Strategy 5 Item Cold-Start Module (Item Groups)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - list of users to recommend the new item
$\vartheta$ - selected items features and contextual information
$\rho$ - frequent itemsets per itemGroup
$\theta$ - acceptable support threshold
$\delta$ - acceptable participation percentage threshold
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho$ $\leftarrow$ fp-growth(dataTrainSets, $\theta$)
for each frequent_itemsets in $\rho$(itemGroup) do
    for each user do
        if user_participation $> \delta$ then
            $Res$(itemGroup) $\leftarrow$ user
        end if
    end for
end for
Recommendation:
$\gamma$ $\leftarrow$ findItemGroup(new_item)
Recommend $new\_item$ to all users in $Res(\gamma)$
```

Firstly, Strategy 1 has a linear time complexity of $O(n_i)$, since it focuses on discovering itemsets of size 1. In Strategy 2, we split the input data using selected users' features, and for each split, we run the FP-growth algorithm to find frequent 1-itemsets, so its time complexity is also $O(n_i)$. In the first step of Strategy 3, FP-growth requires quadratic time to find all frequent usersets of size $\leq k$. So, the time complexity should be $O(n_u^2)$. In the next step, we check the participation percentage for every item that is highly rated by frequent usersets. Proposition 1 shows that the maximum number of frequent usersets of any size $\leq k$ that could be returned by the FP-growth algorithm is lower than $\sum_{k=1}^{n} \frac{n^k}{k!}$. By applying Proposition 2, we can determine the asymptotic equivalence between $\sum_{k=1}^{n} \frac{n^k}{k!}$ and $\frac{n^k}{k!}$. Therefore, the overall time complexity of Strategy 3 in the worst-case scenario is $O(n_u^2) + O(\frac{n_u^k}{k!})O(n_i) = O(n_u^k n_i)$ (cf. Corollary 1). Strategy 4 is the same as Strategy 3 except that it iterates only over the frequent usersets in the dominant group. Hence, its complexity is $O(\frac{1}{c}n_u^k)O(\frac{1}{c}n_i) = O(n_u^k n_i)$, where c is a

Strategy 6 Item Cold-Start Module (Dominant Groups)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - list of users to recommend the new item
$\vartheta$ - selected items and users features and contextual information
$\rho$ - frequent itemsets per userItemGroup
$\theta$ - acceptable support threshold
$\delta$ - acceptable participation percentage threshold
$\beta$ - dominant frequent itemsets per itemGroup (largest itemsets)
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho \leftarrow$ fp-growth(dataTrainSets, $\theta$)
$\beta \leftarrow$ findDominantGroups($\rho$)
for each frequent_itemsets in $\beta$(itemGroup) do
    for each user do
        if user_participation $> \delta$ then
            $Res$(itemGroup) $\leftarrow$ user
        end if
    end for
end for
Recommendation:
$\gamma \leftarrow$ findItemGroup(new_item)
Recommend $new\_item$ to all users in $Res(\gamma)$
```

Strategy 7 Item Cold-Start Module (Clustering-Based)

```
Data:
$dataTrain$ - entire training dataset
$Res$ - list of users to recommend the new item
$\vartheta$ - selected items and users features and contextual information
$\rho$ - frequent 1-itemsets per userItemGroup (clusters)
$\theta$ - acceptable support threshold
$\delta$ - acceptable participation percentage threshold
Procedure:
dataTrainSets $\leftarrow$ split(dataTrain, $\vartheta$)
$\rho \leftarrow$ fp-growth(dataTrainSets, $\theta$)
for each frequent_itemsets in $\rho$(cluster) do
    for each user do
        if user_participation $> \delta$ then
            $Res$(cluster) $\leftarrow$ user
        end if
    end for
end for
Recommendation:
$\gamma \leftarrow$ findClosestCluster(new_item)
Recommend $new\_item$ to all users in $Res(\gamma)$
```

constant representing the number of groups after splitting the dataset. Although the time complexity remains consistent between Strategy 3 and Strategy 4, in practical scenarios where both strategies discover a similar number of frequent

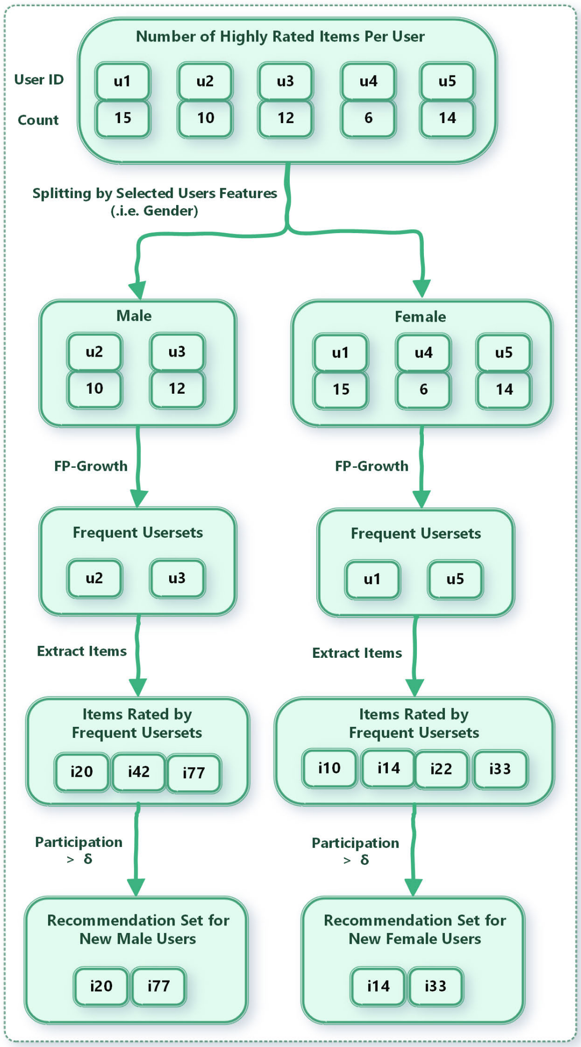

Number of Highly Rated Items Per User
User ID
Count
u1
u2
u3
u4
u5
15
10
12
6
14
Splitting by Selected Users Features (.i.e. Gender)
Male
Female
FP-Growth
Frequent Usersets
Extract Items
Items Rated by Frequent Usersets
i20
i42
i77
i10
i14
i22
i33
Participation > δ
Recommendation Set for New Male Users
Recommendation Set for New Female Users

FIGURE 2. Example of using gender feature in Strategy 3 to address the user cold-start problem.

sets, Strategy 4 can significantly accelerate its execution by iterating solely over the elements in the dominant group.

In the item cold-start module, Strategy 5 mirrors Strategy 3 and Strategy 6 mirrors Strategy 4, indicating a time complexity of $O(n_i^k n_u)$, as previously outlined. In Strategy 7, we find only frequent 1-itemsets for each split, a process that operates in $O(n_i)$ and yields no more than the total number of items. Then, for each item, we identify all users whose participation percentage exceeds the specified threshold. In the worst-case scenario, this strategy would have a runtime of $O(n_i) + O(n_i)O(n_u) = O(n_i n_u)$.

In all the preceding formulas, we assume the worst-case scenario; however, in practice, the strategies execute considerably faster. It is worth noting that, in our experiments, optimal results were consistently achieved with $k = 4$ in all strategies where the generation of frequent sets with size > 1 is required. In Section V-E, we practically demonstrate that for relatively small values of k, the achieved complexity is polynomial rather than exponential, which observation is in line with Corollary 1. Moreover, the performance of the strategies may differ between datasets, and selecting the best one, as well as searching for optimal values for parameters and thresholds, should be subjected to experimental evaluation per case. More details on the implementation of parameter tuning are provided in Section V.

Proposition 1: Let n be the total number of elements, and let k be the size of frequent set. Then, the number of frequent sets returned by FP-growth algorithm with size $\leq k$ is lower than $\sum_{k=1}^{n} \frac{n^k}{k!}$.

Proof: The formula for the number of possible combinations when k objects are selected out of n different objects is given by [73]:

$$^{n}C_k = \binom{n}{k} = \frac{n!}{(n-k)!k!} \tag{13}$$

We can express $^{n}C_k$ in sigma form (or summation form) as follows:

$$\begin{aligned} ^{n}C_1 &= n \\ ^{n}C_2 &= \frac{n(n-1)}{2!} \\ ^{n}C_3 &= \frac{n(n-1)(n-2)}{3!} \end{aligned} \tag{14}$$

When k objects are selected out of n different objects, we can infer the maximal number of frequent sets generated by FP-growth algorithm as follows:

$$\begin{aligned} ^{n}C_k &= \frac{n(n-1)(n-2)\ldots(n-k+1)}{k!} \\ ^{n}C_k &< \frac{n^k}{k!} \end{aligned} \tag{15}$$

Finally, when the size of frequent itemsets is $\leq k$, the maximal number of frequent sets of n different objects can be deduced as follows:

$$\sum_{k=1}^{n} {^{n}C_k} < \sum_{k=1}^{n} \frac{n^k}{k!} \tag{16}$$

■

Proposition 2: Let k be limited by a constant C_k, then:

$$\sum_{k=1}^{C_k} \frac{n^k}{k!} \sim \frac{n^{C_k}}{C_k!}$$

Proof: Since the sum is finite, i.e, limited by C_k:

$$\begin{aligned} &\lim_{n\to\infty} \frac{\sum_{k=1}^{C_k} \frac{n^k}{k!}}{\frac{n^{C_k}}{C_k!}} \\ &= \lim_{n\to\infty} \frac{\frac{n^{C_k}}{C_k!}}{\frac{n^{C_k}}{C_k!}} + \frac{\sum_{k=1}^{C_k-1} \frac{n^k}{k!}}{\frac{n^{C_k}}{C_k!}} \\ &= \lim_{n\to\infty} 1 + \frac{\sum_{k=1}^{C_k-1} \frac{n^k}{k!}}{\frac{n^{C_k}}{C_k!}} = 1 + 0 = 1 \end{aligned} \tag{17}$$

■

Corollary 1: Let k be limited by a constant C_k, then:

$$\frac{n^k}{k!} = \Theta(n^k)$$

Proof: Having k limited by a constant value C_k, we can easily show as n approaches infinity $lim_{n\to\infty}$ $\frac{\frac{n^k}{k!}}{n^k}$ approximates towards a constant. ■

V. EXPERIMENTAL EVALUATION

In this section, we conduct comprehensive experiments to evaluate the performance of the Clustering-based FPRS recommender system.

A. DATASETS AND EVALUATION MEASURES

In our experiments, we used four datasets, namely: MovieLens 100K, MovieLens 1M,[1] MovieLens + IMDb/Rotten Tomatoes (abbreviated as HetRec-MovieLens),[2] and LDOS-CoMoDa.[3] MovieLens datasets were collected by the GroupLens research project at the University of Minnesota. MovieLens 100K contains 100 000 ratings given by 943 users on 1 682 movies on a scale from 1 to 5. While MovieLens 1M includes 1 000 000 ratings of approximately 3 900 films made by 6 040 users on a scale from 1 to 5. Also, we used HetRec-MovieLens dataset, which is an extension of the MovieLens 10M dataset that was published by the GroupLeans research group in 2011 at the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems [74]. It links the movies of the MovieLens dataset with their corresponding web pages at Internet Movie Database (IMDb) and Rotten Tomatoes movie review systems. It contains 2 113 users and 855 598 ratings (on a scale from 0 to 5) of about 10 197 movies. Finally, we employed the LDOS-CoMoDa dataset, which is presented by Kosir et al. [75]. LDOS-CoMoDa is a context-rich movie recommender dataset comprising 200 users who gave 2 296 ratings (on a scale from 0 to 5) for 4 138 movies in twelve pieces of contextual information describing the situation in which the user consumed the item.

In all datasets, we combine *users.data*, *items.data*, and *ratings.data* to join users' characteristics (e.g., demographics), items' attributes (e.g., genre), and ratings in one dataset. The final/merged dataset contains userId, itemId, rating, gender, age, occupation, and genre attributes (cf. Table 1). However, the HetRec-MovieLens dataset does not contain any user characteristics. Instead, it provides the time at which the rating was given. Using this contextual information, we represent each user by a vector where each value corresponds to a pair of (season, time of day) and represents the number of highly rated movies at that time. After normalizing these values, we use the agglomerative clustering technique with Euclidean distance to group the users. Following this approach, we evaluate all strategies of

[1]https://grouplens.org/datasets/movielens/
[2]https://grouplens.org/datasets/hetrec-2011/
[3]https://www.lucami.org/en/research/ldos-comoda-dataset/

the item cold-start module on the HetRec-MovieLens dataset. However, it is not possible to utilize this dataset to evaluate the strategies of the user cold-start module due to missing ratings and contextual information for new users.

Table 1 presents an analysis of selected features used in our experiments (i.e., gender and genre). This perspective lets us better understand the interrelations between features that could impact the results. Figures 3 show the most popular movie genres among males and females for three datasets where genre and gender are available. It is worth noting that the data is slightly imbalanced, e.g., there are more males than females in the datasets.

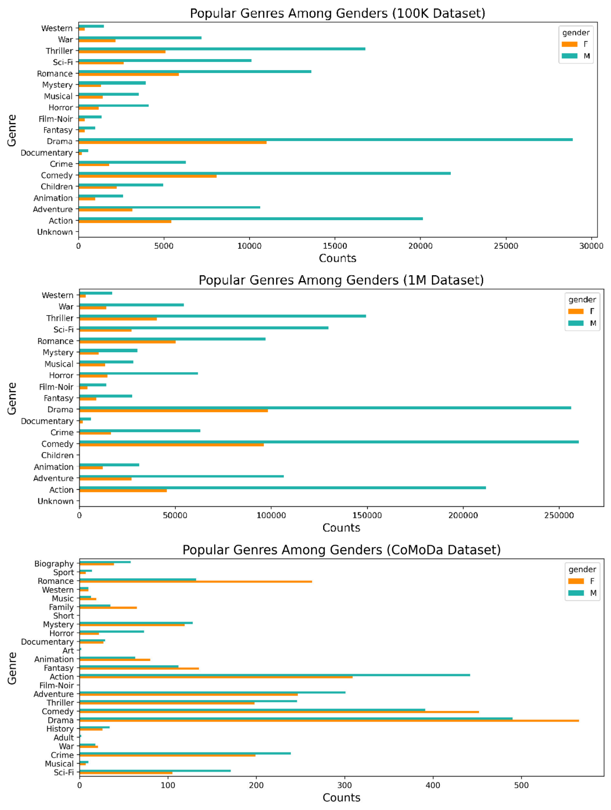


FIGURE 3. Histogram of the variables (rating, genre, and gender) in MovieLens (100K and 1M) and LDOS-CoMoDa datasets.

The data were split into training and testing sets in a way allowing to ensure a sufficient number of ratings in the test set per each gender and genre. In particular:

- For the experiment related to the user cold-start module, we draw some users with a possibly high number of ratings on the 50 popular movies. All the ratings given by all those users (i.e., 9 052 records in MovieLens 100K, 25 533 records in MovieLens 1M, and 519 records in LDOS-CoMoDa) are considered a testing set, keeping the rest of the records in the training set. The number of users is selected in a way such that 20-30% of the records is kept for testing, and the rest is used for training. This way, all the records related to the selected users were removed from the test set, which properly simulates the cold-start problem associated with new users (i.e., no available rating history).
- To properly evaluate the item cold-start module, the testing data is chosen in a very similar way. First, we have drawn 50 active users (i.e., with a considerable number of ratings). Subsequently, we have drawn out some of the most-rated movies by those 50 users. Similarly, the number of items differs between the datasets in order to fairly split the dataset between training and testing. The ratings of all the selected movies by all users (i.e., 7 320 records in MovieLens 100K, 41 105 records in MovieLens 1M, 25 583 records in HetRec-MovieLens, and 209 records in LDOS-CoMoDa) are considered a testing set, keeping the rest of the records in the training set. Note that all the ratings related to the test movies were removed from the training dataset, which models the item cold-start problem well.

In our study, we consider a binary decision task, i.e., whether a given item (e.g., a movie) is appropriate for the user (or not). To correctly model this situation for the MovieLens data, we assume users prefer all the films rated 3 or more (belonging to the positive class). In contrast, those ranked lower are poorly matched to the users. In the described scenario, the Clustering-based FPRS returns binary information for each new user (or item) - to recommend or not. Following that, to assess the quality of the prediction, the F1-score is used [76]:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{18}$$

where precision and recall, which is also known as True Positive Rate (TPR), are well-known binary evaluation metrics, cf. Equations 19 and 20.

$$Precision = \frac{TP}{TP + FP} \tag{19}$$

$$Recall\ (True\ Positive\ Rate) = \frac{TP}{TP + FN} \tag{20}$$

We also use accuracy (Equation 21) to measure all the correctly identified cases. This measure is most effective for the data with balanced decision classes that are equally important. However, the accuracy measure is helpful to gain some insights about the error rate since they are inversely related *(Error Rate = 1 − Accuracy)*.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{21}$$

Finally, we use false positive rate (FPR) to report the proportion of negative cases incorrectly identified as positive cases, cf. Equation 22.

$$False\ Positive\ Rate\ (FPR) = \frac{FP}{FP + TN} \tag{22}$$

B. BASELINE RECOMMENDER SYSTEMS

To show the strengths of Clustering-based FPRS, we evaluate it against state-of-the-art models which are designed to mitigate the cold-start problem in recommender systems.

TABLE 1. Selected data characteristics.

Attribute Name	Data Type	Value Range (MovieLens 100K)	Value Range (MovieLens 1M)	Value Range (HetRec-MovieLens)	Value Range (LDOS-CoMoDa)
number of users	NA	943	6,040	2113	200
number of items	NA	1682	3,900	10197	4138
number of rows (ratings)	NA	100,000	1000,000	855,598	2296
rating density	NA	6.30%	4.25%	3.97%	0.28%
gender	Character	M-F	M-F	NA	M-F
age	Number	Under 18-73	Under 18-56	NA	15-63
occupation	Text	21 occupation	21 occupation	NA	NA
genre	Text	19 genres	19 genres	20 genres	25 genres
season (context)	Text	NA	NA	4 seasons	4 seasons
time (context)	Text	NA	NA	4 values	4 values

The first baseline [20], called regularization differentiating functions (RDF), was proposed recently to alleviate the cold-start problem by using three simple regularization differentiating functions on top of the matrix factorization-based models. The main objective behind using these functions is to set lower regularization weights to the latent factors associated with popular items and active users, and assign higher regularization weights on unpopular, less rated/viewed/purchased, items and less active users to prevent their latent factors from being influenced by the few extreme observations. In particular, this method aims to improve the MF-based models by utilizing more information revealed by popular items and active users, and making conservative predictions on unpopular items and less active users. In the evaluation, we used the publicly available implementation,[1] provided by the authors, which integrates the proposed regularization technique with the SVD, SVD++, and the NMF models.

The second baseline [21], called collective matrix factorization models in recommender systems (CMFREC), proposed an enhancement to the collective matrix factorization model to deal with binary data. In addition, it provided an alternative formulation, called offsets model, that is able to make fast recommendations for new users and items and does not require any transformation for attributes data that is limited in range. On the other hand, we also evaluate our strategies against two classical and well-known latent factor models, namely Matrix Factorization (MF) [43] and Singular Value Decomposition (SVD) [5], which both rely on transforming items and users to the same latent factor or embedding space. The embedding space, also known as item- and user-embeddings, aims to explain ratings by finding the dependencies between rows (users) and columns (items) in the user-item rating matrix. Eventually, these embeddings are used to predict the missing ratings in the original matrix. It is worth noting that the implementations of MF and SVD provided by the Cornac library have been used in the evaluation process [77].

[1]https://github.com/ncu-dart/rdf

Furthermore, to measure the impact of frequent pattern mining in RS, we developed two versions of a baseline model similar to the strategies described previously, in which, however, the FP-growth algorithm was omitted, as follows:

- *Baseline* model for new users: for each movie in the training set, the most common value of the selected user's feature (such as gender and age group) was found. For a given new user, all movies assigned to their age group and gender (or any other selected user characteristic) are recommended.
- *Baseline* model for new items: it works similarly. Here, we pick the most popular value of the selected item's feature for each user. For instance, if we consider genre, each new movie is recommended to all users whose favorite genre (i.e., with the highest ratings in the past) was the same as the new movie's genre.

C. PERFORMANCE COMPARISON AND ANALYSIS

To provide a reliable evaluation of the developed framework, we rely on several metrics, particularly precision, recall, F1, accuracy, and false positive rate (FPR) measures. The two experiments evaluating user cold-start and item cold-start modules were conducted separately, and the outcomes are provided in the subsequent sections.

In Tables 2, 3, and 4, we find precision, recall, F1, accuracy, and FPR measures for the results generated by Baseline models and strategies constituting Clustering-based FPRS dedicated to addressing user cold-start, which are illustrated in detail in Section IV-A. The highest precision, recall, F1, and accuracy values and the lowest false positive rate are bold.

The results confirm that all developed strategies outperform the benchmark models, namely Baseline, RDF, CMFREC, MF, and SVD, in terms of precision, F1, accuracy, and false positive rate on all evaluated data sets. The benchmark solutions reported higher recall which is quite natural since these methods are more selective, providing more apt results with a tradeoff that some potentially relevant movies may be omitted. However, the higher false positive rates reported by the benchmark models indicate that these models made many irrelevant recommendations. For the user cold-start problem, all strategies applied to MovieLens 100k and MovieLens 1M were evaluated at *min_support* value of 0.08 and performed similarly. However, the third one was just slightly better. All strategies evaluated on the LDOS-CoMoDa dataset performed best at *min_support* value of 0.06.

TABLE 2. Evaluation for user cold-start (MovieLens 100k).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.24	0.72	0.33	0.44	0.66
RDF	0.28	**0.75**	0.41	0.82	0.46
CMFREC	0.27	0.68	0.38	0.57	0.46
MF	0.27	0.70	0.39	0.58	0.44
SVD	0.28	0.71	0.40	0.58	0.45
Strategy 1	0.58	0.56	0.57	0.80	0.09
Strategy 2	**0.59**	0.58	0.58	0.80	0.09
Strategy 3	0.57	0.69	**0.61**	**0.87**	**0.06**
Strategy 4	0.54	0.70	0.59	0.83	**0.06**

TABLE 3. Evaluation for user cold-start (MovieLens 1M).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.33	**0.82**	0.44	0.47	0.77
RDF	0.33	0.64	0.37	0.61	0.35
CMFREC	0.30	0.65	0.40	0.59	0.42
MF	0.30	0.67	0.41	0.59	0.42
SVD	0.31	0.68	0.41	0.60	0.42
Strategy 1	0.63	0.63	0.62	0.77	0.04
Strategy 2	**0.64**	0.64	0.63	0.78	0.05
Strategy 3	0.59	0.73	**0.67**	**0.85**	**0.03**
Strategy 4	0.59	0.55	0.56	0.72	0.17

In the second experiment, we evaluated the item cold-start module of Clustering-based FPRS. We calculated precision, recall, F1, accuracy, and false positive rate measures for the results generated by Baseline, RDF, CMFREC, MF, SVD, and all the strategies described in Section IV-A. The comparative summary of this evaluation is shown in Tables 5, 6, 7 and 8.

The results confirm that the developed approach to mitigate the item cold-start problem is superior to the baseline solution. However, the observations differ slightly between datasets. For MovieLens 100K, all strategies reported similar recall. Regarding precision and F1 measures, the most successful in dealing with new items in this data appeared to be Strategy 6, which is based on items' and users'

TABLE 4. Evaluation for user cold-start (LDOS-CoMoDa).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.08	0.23	0.11	0.92	0.41
RDF	0.03	**0.48**	0.04	0.63	0.20
CMFREC	0.02	0.35	0.04	0.80	0.19
MF	0.04	0.36	0.05	0.82	0.16
SVD	0.03	0.39	0.04	0.80	0.19
Strategy 1	**0.57**	0.05	0.10	**0.97**	0.04
Strategy 2	0.39	0.06	0.09	**0.97**	**0.02**
Strategy 3	0.17	0.22	**0.16**	0.96	0.05
Strategy 4	0.25	0.11	0.15	**0.97**	0.06

characteristics. However, for the applications that do require high accuracy and low false positive rate (FPR), it may be better to apply Strategy 7, which was also superior in terms of recall, F1, accuracy, and FPR on the MovieLens 1M data. Still, for the LDOS-CoMoDa and HetRec datasets, the best F1-score, accuracy, and FPR were reported by Strategy 6. When it comes to benchmark models, we observe that Baseline showed higher accuracy than other models. However, all these models reported relatively similar F1 scores.

It is noteworthy that to select the proper strategy, we should rely on the appropriate evaluation measure that best matches the target of our application. For instance, we should count on precision scores when our application focuses on the number of correct recommendations considering the mistakes made. While it is preferable to rely on recall scores when the targeted application is not concerned with the mistakes made since the recall measure only considers the number of correct recommendations made out of all positive predictions that could have been made (cf. Equations 19 and 20).

On MovieLens datasets (100k and 1M), all strategies were evaluated at the participation threshold value of 30% and *min_support* value of 0.22. On the HetRec-MovieLens dataset, all strategies were evaluated at a threshold value of 15% and *min_support* value of 0.20. Lastly, on the LDOS-CoMoDa, the best threshold and *min_support* values were 20% and 0.06, respectively. In all experiments, we trained the models using the optimal values of thresholds. More details about tuning the thresholds are presented in Section V-D.

Let us now take a closer look at the performance of the proposed strategies with the Receiver Operating Characteristic (ROC) curves. ROC curves plot the True Positive Rate (TPR) against False Positive Rate (FPR) at various threshold values (cf. Equations 20 and 22) and are often applied to evaluate how well logistic regression models classify positive vs. negative classes at all possible cutoffs. The Area Under the ROC Curve (AUC) is typically used as a numeric summary of the ROC curve by measuring the ability of a binary classifier to distinguish or separate between the positive and negative classes [76].

TABLE 5. Evaluation for item cold-start (MovieLens 100k).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.38	0.3	0.32	0.59	0.26
RDF	0.35	0.51	0.41	0.48	0.53
CMFREC	0.33	0.46	0.38	0.47	0.52
MF	0.35	0.49	0.37	0.49	0.50
SVD	0.36	0.50	0.35	0.50	0.50
Strategy 5	0.69	**0.64**	0.66	0.75	0.22
Strategy 6	**0.79**	0.62	**0.69**	0.84	0.12
Strategy 7	0.67	0.63	0.63	**0.86**	**0.10**

TABLE 6. Evaluation for item cold-start (MovieLens 1M).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.49	0.43	0.43	0.64	0.37
RDF	0.34	0.52	0.39	0.51	0.49
CMFREC	0.30	0.46	0.36	0.48	0.52
MF	0.32	0.49	0.38	0.50	0.50
SVD	0.31	0.49	0.37	0.49	0.50
Strategy 5	**0.65**	0.71	0.64	0.76	0.29
Strategy 6	0.64	0.73	**0.66**	0.83	0.16
Strategy 7	0.6	**0.79**	**0.66**	**0.86**	**0.13**

TABLE 7. Evaluation for item cold-start (LDOS-CoMoDa).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.07	**0.78**	0.14	0.64	0.69
RDF	0.07	0.59	0.13	0.42	0.58
CMFREC	0.04	0.20	0.07	0.46	0.51
MF	0.13	0.60	0.21	0.53	0.47
SVD	0.14	0.57	0.22	0.58	0.43
Strategy 5	0.39	0.58	0.43	0.92	0.05
Strategy 6	**0.61**	0.54	**0.51**	**0.98**	**0.01**
Strategy 7	0.38	0.28	0.31	0.96	0.03

TABLE 8. Evaluation for item cold-start (HetRec-MovieLens).

Strategy	Precision	Recall	F1	Accuracy	FPR
Baseline	0.39	0.58	0.45	0.55	0.70
RDF	0.56	0.55	0.55	0.52	0.48
CMFREC	0.52	0.47	0.48	0.47	0.54
MF	0.53	0.48	0.50	0.48	0.52
SVD	0.53	0.48	0.50	0.48	0.52
Strategy 5	0.72	**0.88**	0.77	0.74	0.23
Strategy 6	0.74	0.87	**0.79**	**0.92**	**0.16**
Strategy 7	**0.76**	0.56	0.60	0.72	0.27

In the experiment, we utilize AUC/ROC score to find the most efficient model in the user cold-start module. The Clustering-based FPRS is a binary decision recommender system, i.e., provides a binary decision "to recommend" (1) or "not" (0), instead of ranking (e.g., with logistic regression). Therefore, to generate ROC curves, we slightly modified the implemented algorithms. In particular, we rank the generated recommendations using the support value in Strategies 1 and 2, and the participation percentage value in Strategies 3 and 4. The recommendations generated by benchmark models are ranked/ordered based on their estimated ratings. There is one exception, we could not calculate AUC/ROC score for the *Baseline* model since it is a binary classifier that predicts True/False for every instance in the test set without prior estimating of probability.

Figures 4a and 4b plot ROC curves achieved at different threshold values for every model using the most representative datasets, i.e., MovieLens 100k and 1M. Table 9 displays AUC scores for all ROC curves. The results confirm that all developed strategies outperform the benchmark models. However, they reported relatively similar AUC scores for the examined datasets. For the MovieLens 100k data, the Strategies 1 and 2 achieved the highest AUC of 0.85 for new/cold users. Whereas, for the MovieLens 1M data, the Strategy 3 was superior, with an AUC of 0.87.

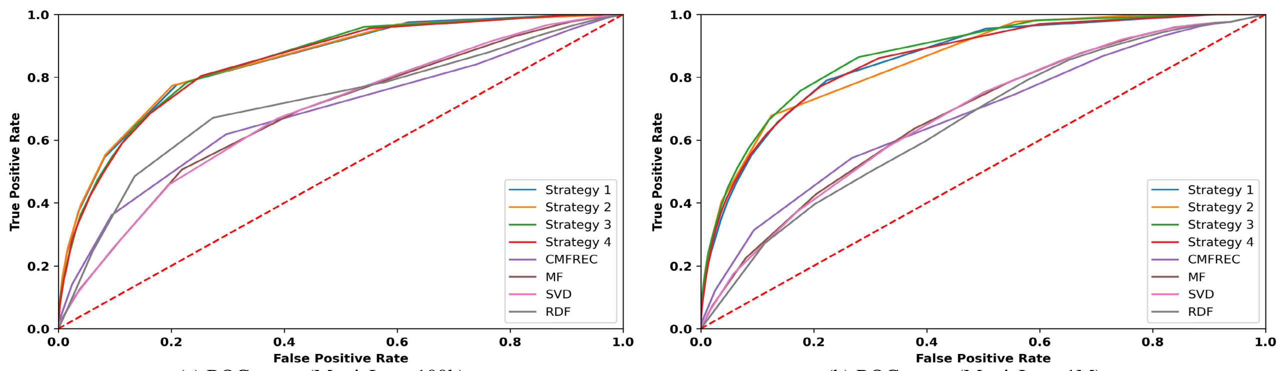
TABLE 9. AUC values for each ROC curve shown in Figure 4.

ROC Curve	MovieLens100k	MovieLens1M
RDF	0.71	0.64
CMFREC	0.68	0.67
MF	0.68	0.68
SVD	0.69	0.67
Strategy 1	**0.85**	0.86
Strategy 2	**0.85**	0.85
Strategy 3	0.84	**0.87**
Strategy 4	0.84	0.86

D. THRESHOLDS SENSITIVITY ANALYSIS

The performance of the Clustering-based FPRS model depends on *min_support* and participation thresholds, which mainly impact the extraction of frequent itemsets and relevant recommendations for new users and new items. In this section, we elaborate on the impact of parameter tuning on our model performance. Besides, additional experiments were conducted to find the optimal number of user and item clusters to model their similarity. The outputs of this study establish the optimal values of these thresholds.

In the first experiment, we aim to find the optimal value of *min_support* threshold by evaluating the item cold-start module of the Clustering-based FPRS framework using different *min_support* values. Figure 6a shows how



(a) ROC curve (MovieLens 100k) (b) ROC curve (MovieLens 1M)

FIGURE 4. Roc curves for Clustering-based FPRS strategies (user module) and benchmark models using MovieLens (100k and 1M) datasets.

the F1-score is impacted by applying different values for MovieLens 100K data. Observably, the best *min_support* values for all strategies in the item cold-start module lay between 0.1 and 0.2. A similar analysis, yet regarding MovieLens 1M, HetRec-MovieLens, and LDOS-CoMoDa data sets, we may find in Figures 6b, 6c, and 6d, respectively.

In the second experiment, we plot the impact of the participation threshold on the F1-score for the strategies in the item cold-start module. Figures 6e, 6f, 6g, and 6h plot the F1-score achieved for different threshold values for the examined datasets. Observably, the best participation threshold values for all strategies used in FPRS (item cold-start module) are around 20% for MovieLens (100K and 1M), 5% for HetRec-MovieLens and 10% for LDOS-CoMoDa. Finally, it is worth noting that when we run this experiment, we apply the optimal value of *min_support* from the previous investigation.

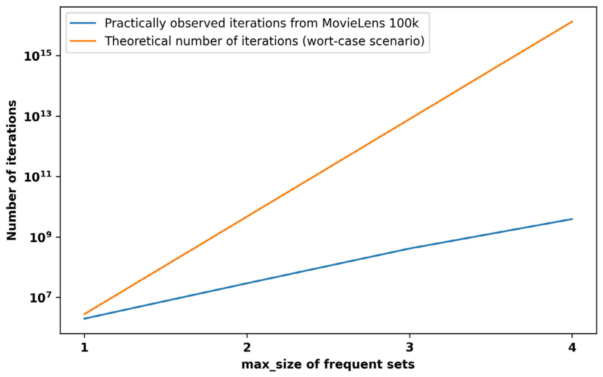


FIGURE 5. Comparing the practical time complexity of Strategy 4 with the worst-case scenario outlined in Section IV-A.

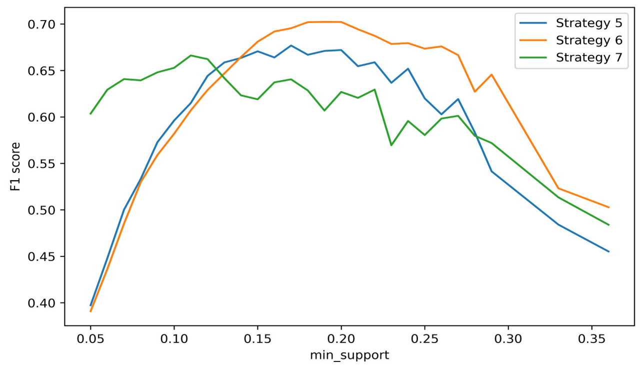
Finally, we test the optimal number of clusters to model users' and items' similarity. For that purpose, each item is represented by a vector of genres. In our study, we employ agglomerative clustering. In this technique, each observation starts as a singleton cluster. In the following steps, clusters are recursively merged, minimizing the maximum distance between pairs of items. Two clusters are merged at level *c* if every possible pair of their elements has a distance smaller than this level. Searching for the optimal number of clusters (that correspond to searching for the optimal *c*), we aimed at achieving a possibly small number of clusters, i.e., a possibly high value *c*, which was ultimately established experimentally at the level of 0.9. In the experiment, we used Jaccard distance, which is very convenient for comparing observations with categorical variables.

TABLE 10. Time taken for model training in user cold-start module (in seconds).

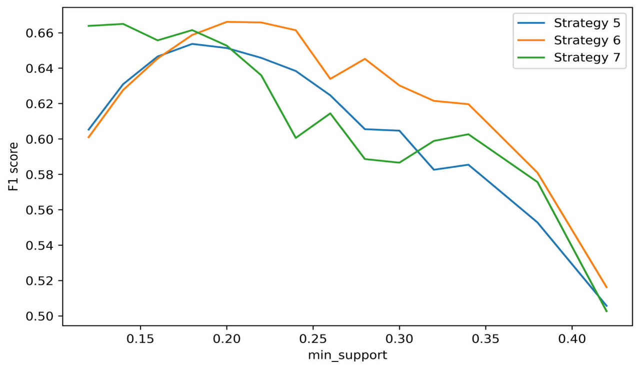
Strategy	Movie-Lens100k	Movie-Lens1M	LDOS-CoMoDa
Baseline	21.06	212.28	0.64
RDF	112.73	1119.93	15.69
CMFREC	8.27	10.60	4.28
MF	0.24	2.86	0.03
SVD	0.09	0.73	0.02
Strategy 1	1.89	11.99	0.08
Strategy 2	2.48	14.18	0.15
Strategy 3	0.99	3.22	0.73
Strategy 4	2.37	6.24	1.94

TABLE 11. Time taken for model training in item cold-start module (in seconds).

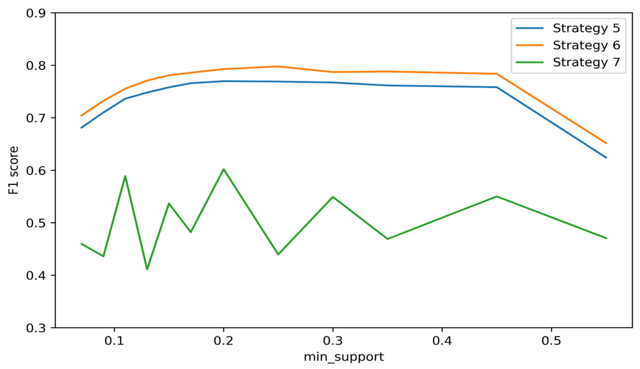
Strategy	Movie-Lens100k	Movie-Lens1M	LDOS-CoMoDa	HetRec-MovieLens
Baseline	15.84	162.48	2.07	114.28
RDF	114.65	1107.38	7.59	962.84
CMFREC	5.14	10.58	4.33	9.87
MF	0.31	2.72	0.03	2.78
SVD	0.11	0.50	0.003	0.53
Strategy 5	1.45	10.15	0.61	4.92
Strategy 6	1.85	11.16	3.2	5.41
Strategy 7	2.49	10.07	1.61	7.45



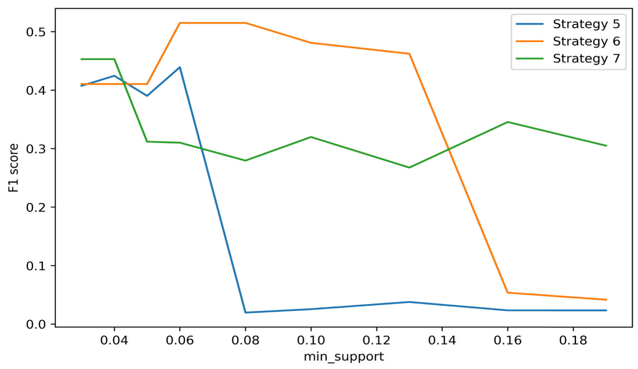
(a) min_supp threshold (MovieLens 100k)



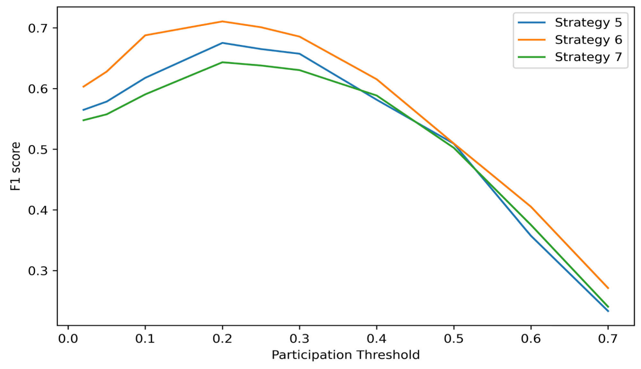
(b) min_supp threshold (MovieLens 1M)



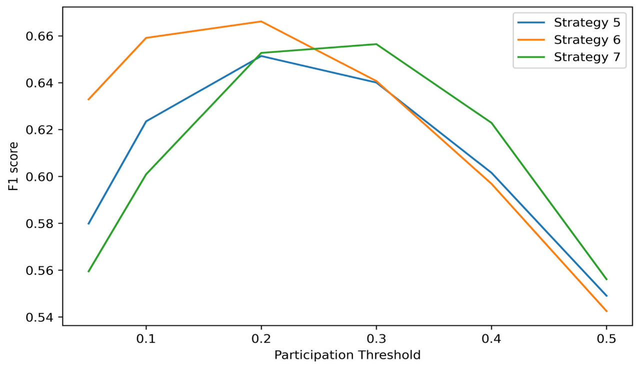
(c) min_supp threshold (HetRec-MovieLens)



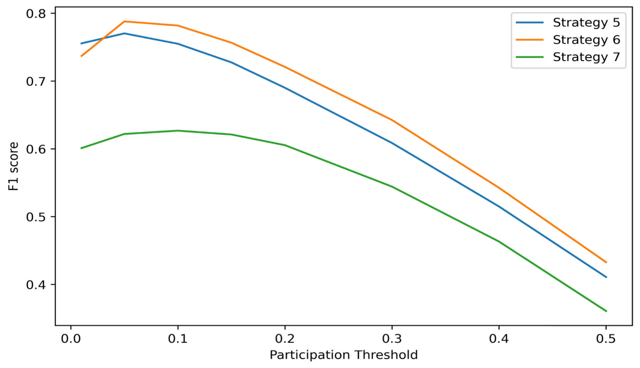
(d) min_supp threshold (LDOS-CoMoDa)



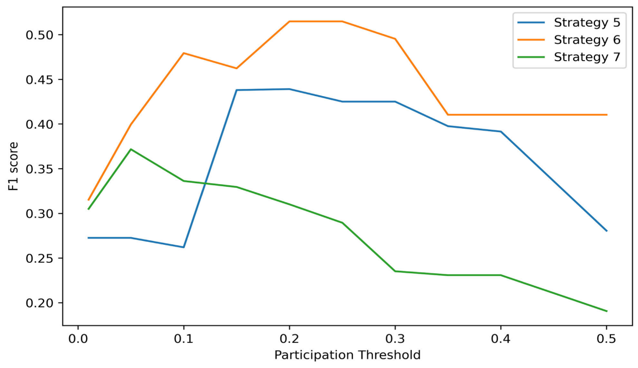
(e) participation percentage threshold (MovieLens 100k)



(f) participation percentage threshold (MovieLens 1M)



(g) participation percentage threshold (HetRec-MovieLens)



(h) participation percentage threshold (LDOS-CoMoDa)

FIGURE 6. Sensitivity analysis for min_supp and participation percentage thresholds in MovieLens (100k and 1M), HetRec-MovieLens and LDOS-CoMoDa.

E. RUN-TIME COMPARISONS

The primary objective of this section is to assess the efficiency of our strategies in terms of computational complexity. To illustrate, we opt for Strategy 4 and practically evaluate its complexity against the worst-case scenario (cf. Section IV-A). Figure 5 displays the observed number of iterations while running Strategy 4 on the MovieLens 100K dataset (blue line) alongside the theoretically worst-case number of iterations (orange line). The results demonstrate that the practical complexity is significantly lower than the worst-case scenario. This discrepancy arises because the number of frequent sets returned by the FP-growth algorithm is considerably smaller than the maximum number indicated in Proposition 1. This confirms that selecting a reasonably small value for k (or *max_size* of frequent sets) results in polynomial complexity rather than exponential. In Figure 5, we use a logarithmic scale in order to display data with expansive value disparity on the same chart.

On the other hand, we observed that fine-tuning the *min_support* parameter is crucial to achieve the best results. Our experiments with various datasets of different sizes demonstrate that the optimal value of *min_support* consistently restricts the FP-growth algorithm from discovering frequent sets with large sizes. For instance, after optimizing the *min_support*, we consistently observed the best results when setting $k = 4$ in all strategies where the generation of frequent sets with size > 1 is required.

To show the practical impact of previous discussions, we compare the developed strategies with all baselines in terms of the computational time of training the model on all datasets. The results in Tables 10 and 11 show that the time required to train the models in Clustering-based FPRS framework is significantly shorter compared to baseline models in user and item cold-start modules. When it comes to generating cold-start recommendations, Clustering-based FPRS models require less than 0.3 seconds to generate recommendations for a new user or to recommend a new item to some users which makes the proposed framework suitable for real-time systems.

F. CHALLENGES AND LIMITATIONS

This section is dedicated to discussing the limitations of our proposed method and the challenges we faced while evaluating it. The most important among which is the availability of users' and items' characteristics which are used to split the dataset and extract specified patterns for each subset of the dataset. However, we can utilize the contextual information instead of users' and items' features when they are missing.

Another issue appears when we deal with small datasets. It is crucial to have enough transactions in every split in order to extract frequent itemsets. It would be ideal to have a similar number of transitions in every subset, however, if this is not the case, then we have to carefully select appropriate values for input parameters, such as *min_support* and *participation* percentage thresholds, by analyzing the results of the experiments presented in Section V-D. This is a bit challenging especially when we deal with small datasets.

VI. CONCLUSION AND FUTURE WORKS

This article presents Clustering-based FPRS - a recommender system that utilizes ratings to discover frequent itemsets associated with selected users/items features and then incorporates these frequent itemsets in generating recommendations for new users and items. The developed clustering-based feature extraction phase aims at modeling similarity between investigated entities. This way, we not only extend the data representation but also increase their density, allowing us to alleviate the omnipresent problem of too few interactions in historical data, especially severe for new products or services. Our study proposes and evaluates multiple strategies for creating frequent itemsets to produce meaningful and relevant recommendations for new users and items. The discovered frequent itemsets are dedicated to specific groups of users and items which are generated based on selected users' and items' characteristics, or based on the ratings given by users, or assigned to items in a dedicated context.

To evaluate Clustering-based FPRS, we conducted experiments on MovieLens (100K and 1M), HetRec-MovieLens, and LDOS-CoMoDa datasets with the FP-growth algorithm to generate the frequent itemsets. The experimental results show that Clustering-based FPRS has outperformed state-of-the-art models, designed to address the cold-start problem, in terms of precision, recall, F_1, accuracy, and FPR measures.

In the future, we plan to investigate more algorithms for association rule and frequent pattern mining, e.g., AprioriTID or Apriori Hybrid. An important extension of the proposed method will be the introduction of additional permissions allowing for incremental processing of large data sets [40], [78] as well as enabling interaction with the user in this process of deriving a granular representation of data [25], [79]. We also plan to respond better to changes in users' behavior and preferences, addressing the possible drifts and shifts in data. One viable option is to periodically update frequent itemsets based on recent changes in rating history. It would also be of value to extend the users' and items' data representation by applying a more advanced feature extraction to model the similarities among them more effectively [80], [81], [82].

We also find it very promising for future research to investigate the application of granular methods for modeling users' and items' similarity [25], [83] and to explore soft computing methods, incl. rough sets, to model more complex dependencies in data [26], [84], [85]. We plan to improve the evaluation process by using specific and more accurate values of the thresholds (min_support and participation percentage) for each strategy instead of using the average of the best threshold values. After augmenting the developed framework with the discussed extensions, it would be also advisable to subject it to in-depth experimental analysis on a number of real data sets from various domains, including the FMCG industry and supply chain management [3], [86]. We believe the developed framework will be equally effective in all those applications.

REFERENCES

[1] M. Tsagkias, T. H. King, S. Kallumadi, V. Murdock, and M. de Rijke, ''Challenges and research opportunities in eCommerce search and recommendations,'' *ACM SIGIR Forum*, vol. 54, no. 1, pp. 1–23, Feb. 2021.

[2] F. Merabet and D. Benmerzoug, ''QoS prediction for service selection and recommendation with a deep latent features autoencoder,'' *Comput. Sci. Inf. Syst.*, vol. 19, no. 2, pp. 709–733, 2022.

[3] M. Grzegorowski, J. Litwin, M. Wnuk, M. Pabis, and L. Marcinowski, ''Survival-based feature extraction—Application in supply management for dispersed vending machines,'' *IEEE Trans. Ind. Informat.*, vol. 19, no. 3, pp. 3331–3340, Mar. 2023, doi: 10.1109/TII.2022.3178547. https://doi.org/10.1109/TII.2022.3178547

[4] P. M. Alamdari, N. J. Navimipour, M. Hosseinzadeh, A. A. Safaei, and A. Darwesh, ''A systematic study on the recommender systems in the e-commerce,'' *IEEE Access*, vol. 8, pp. 115694–115716, 2020.

[5] E. Kannout, M. Grzegorowski, and H. S. Nguyen, ''Toward recommender systems scalability and efficacy,'' in *Concurrency, Specification and Programming*. Cham, Switzerland: Springer, 2023, pp. 91–121.

[6] A. Pawlicka, M. Pawlicki, R. Kozik, and R. S. Choraś, ''A systematic review of recommender systems and their applications in cybersecurity,'' *Sensors*, vol. 21, no. 15, p. 5248, Aug. 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/15/5248

[7] Z. Batmaz, A. Yurekli, A. Bilge, and C. Kaleli, ''A review on deep learning for recommender systems: Challenges and remedies,'' *Artif. Intell. Rev.*, vol. 52, no. 1, pp. 1–37, Jun. 2019.

[8] A. Panteli and B. Boutsinas, ''Addressing the cold-start problem in recommender systems based on frequent patterns,'' *Algorithms*, vol. 16, no. 4, p. 182, Mar. 2023. [Online]. Available: https://www.mdpi.com/1999-4893/16/4/182

[9] M. Kawai, H. Sato, and T. Shiohama, ''Topic model-based recommender systems and their applications to cold-start problems,'' *Expert Syst. Appl.*, vol. 202, Sep. 2022, Art. no. 117129.

[10] M. Pulis and J. Bajada, ''Siamese neural networks for content-based cold-start music recommendation,'' in *Proc. 15th ACM Conf. Recommender Syst.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 719–723.

[11] E. Kannout, M. Grodzki, and M. Grzegorowski, ''Utilizing frequent pattern mining for solving cold-start problem in recommender systems,'' in *Proc. 17th Conf. Comput. Sci. Intell. Syst. (FedCSIS)*, Sofia, Bulgaria, Sep. 2022, pp. 217–226.

[12] E. Kannout, M. Grodzki, and M. Grzegorowski, ''Towards addressing item cold-start problem in collaborative filtering by embedding agglomerative clustering and FP-growth into the recommendation system,'' *Comput. Sci. Inf. Syst.*, vol. 20, no. 4, pp. 1343–1366, 2023.

[13] Z. Tilahun, H. D. Jun, and A. Oad, ''Solving cold-start problem by combining personality traits and demographic attributes in a user based recommender system,'' *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 7, no. 5, pp. 231–239, May 2017.

[14] I. Viktoratos, A. Tsadiras, and N. Bassiliades, ''Combining community-based knowledge with association rule mining to alleviate the cold start problem in context-aware recommender systems,'' *Expert Syst. Appl.*, vol. 101, pp. 78–90, Jul. 2018.

[15] B. Walek and V. Fojtik, ''A hybrid recommender system for recommending relevant movies using an expert system,'' *Expert Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113452.

[16] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro, ''Semantics-aware content-based recommender systems,'' in *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2015, pp. 119–159, doi: 10.1007/978-1-4899-7637-6_4.

[17] H. Lee, J. Im, S. Jang, H. Cho, and S. Chung, ''MeLU: Meta-learned user preference estimator for cold-start recommendation,'' in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1073–1082.

[18] J. Feng, Z. Xia, X. Feng, and J. Peng, ''RBPR: A hybrid model for the new user cold start problem in recommender systems,'' *Knowl.-Based Syst.*, vol. 214, Feb. 2021, Art. no. 106732.

[19] S.-M. Choi, K. Jang, T.-D. Lee, A. Khreishah, and W. Noh, ''Alleviating item-side cold-start problems in recommender systems using weak supervision,'' *IEEE Access*, vol. 8, pp. 167747–167756, 2020.

[20] H.-H. Chen and P. Chen, ''Differentiating regularization weights—A simple mechanism to alleviate cold start in recommender systems,'' *ACM Trans. Knowl. Discovery Data*, vol. 13, no. 1, pp. 1–22, Jan. 2019, doi: 10.1145/3285954.

[21] D. Cortes, ''Cold-start recommendations in collective matrix factorization,'' 2018, *arXiv:1809.00366*.

[22] A. L. V. Pereira and E. R. Hruschka, ''Simultaneous co-clustering and learning to address the cold start problem in recommender systems,'' *Knowl.-Based Syst.*, vol. 82, pp. 11–19, Jul. 2015.

[23] Y. Lu, Y. Fang, and C. Shi, ''Meta-learning on heterogeneous information networks for cold-start recommendation,'' in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 1563–1573, doi: 10.1145/3394486.3403207.

[24] M. Nilashi, O. Ibrahim, and K. Bagherifard, ''A recommender system based on collaborative filtering using ontology and dimensionality reduction techniques,'' *Expert Syst. Appl.*, vol. 92, pp. 507–520, Feb. 2018.

[25] M. Grzegorowski, A. Janusz, D. Ślęzak, and M. S. Szczuka, ''On the role of feature space granulation in feature selection processes,'' in *Proc. IEEE Int. Conf. Big Data (IEEE BigData)*, J. Nie, Z. Obradovic, T. Suzumura, R. Ghosh, R. Nambiar, C. Wang, H. Zang, R. Baeza-Yates, X. Hu, J. Kepner, A. Cuzzocrea, J. Tang, and M. Toyoda, Eds., Boston, MA, USA. Washington, DC, USA: IEEE Computer Society, Dec. 2017, pp. 1806–1815.

[26] M. Grzegorowski and D. Ślęzak, ''On resilient feature selection: Computational foundations of r-$\mathbb{C}$-reducts,'' *Inf. Sci.*, vol. 499, pp. 25–44, Oct. 2019.

[27] M. Grzegorowski, ''Governance of the redundancy in the feature selection based on rough sets' reducts,'' in *Proc. Int. Joint Conf. Rough Sets (IJCRS)*, in Lecture Notes in Computer Science, vol. 9920, V. Flores, F. A. C. Gomide, A. Janusz, C. Meneses, D. Miao, G. Peters, D. Ślęzak, G. Wang, R. Weber, and Y. Yao, Eds., Santiago de Chile, Chile. Cham, Switzerland: Springer, Oct. 2016, pp. 548–557, doi: 10.1007/978-3-319-47160-0_50.

[28] M. Asid and R. Ali, ''Use of soft computing techniques for recommender systems: An overview,'' in *Applications of Soft Computing for the Web*. Singapore: Springer, 2017, pp. 61–80.

[29] Z. Khan, M. I. Hussain, N. Iltaf, J. Kim, and M. Jeon, ''Contextual recommender system for e-commerce applications,'' *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107552.

[30] E. Kannout, ''Context clustering-based recommender systems,'' in *Proc. 15th Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, vol. 21, Sep. 2020, pp. 85–91.

[31] A. Odic, M. Tkalcic, J. F. Tasic, and A. Košir, ''Predicting and detecting the relevant contextual information in a movie-recommender system,'' *Interacting Comput.*, vol. 25, no. 1, pp. 74–90, Jan. 2013.

[32] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, and Y. Zhang, ''Deep reinforcement learning based recommendation with explicit user-item interactions modeling,'' 2018, *arXiv:1810.12027*.

[33] Y. Pérez-Almaguer, R. Yera, A. A. Alzahrani, and L. Martínez, ''Content-based group recommender systems: A general taxonomy and further improvements,'' *Expert Syst. Appl.*, vol. 184, Dec. 2021, Art. no. 115444. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421008587

[34] Z. Zhu, J. Kim, T. Nguyen, A. Fenton, and J. Caverlee, ''Fairness among new items in cold start recommender systems,'' in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.* New York, NY, USA: Association for Computing Machinery, 2021, pp. 767–776.

[35] H. Steck, ''Collaborative filtering via high-dimensional regression,'' 2019, *arXiv:1904.13033*.

[36] S. Natarajan, S. Vairavasundaram, S. Natarajan, and A. H. Gandomi, ''Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data,'' *Expert Syst. Appl.*, vol. 149, Jul. 2020, Art. no. 113248.

[37] H. Kwasnicka and T. Ovedenski, ''Pix2Trips—A system supporting small groups of urban tourists,'' in *Proc. 16th Conf. Comput. Sci. Intell. Syst. (FedCSIS)*, Sep. 2021, pp. 141–145.

[38] H. Sobhanam and A. K. Mariappan, ''Addressing cold start problem in recommender systems using association rules and clustering technique,'' in *Proc. Int. Conf. Comput. Commun. Informat.*, Jan. 2013, pp. 1–5.

[39] W. Feng, Q. Zhu, J. Zhuang, and S. Yu, "An expert recommendation algorithm based on Pearson correlation coefficient and FP-growth," *Cluster Comput.*, vol. 22, no. S3, pp. 7401–7412, May 2019.

[40] E. Kannout, H. S. Nguyen, and M. Grzegorowski, "Speeding up recommender systems using association rules," in *Intelligent Information and Database Systems*. Cham, Switzerland: Springer, 2022, pp. 167–179, doi: 10.1007/978-3-031-21967-2_14.

[41] N. Nassar, A. Jafar, and Y. Rahhal, "A novel deep multi-criteria collaborative filtering model for recommendation system," *Knowl.-Based Syst.*, vol. 187, Jan. 2020, Art. no. 104811.

[42] R. Chen, Q. Hua, Y.-S. Chang, B. Wang, L. Zhang, and X. Kong, "A survey of collaborative filtering-based recommender systems: From traditional methods to hybrid methods based on social networks," *IEEE Access*, vol. 6, pp. 64301–64320, 2018.

[43] Y. Koren, S. Rendle, and R. Bell, *Advances in Collaborative Filtering*. New York, NY, USA: Springer, 2022, pp. 91–142, doi: 10.1007/978-1-0716-2197-4_3.

[44] A. Jain, S. Nagar, P. K. Singh, and J. Dhar, "EMUCF: Enhanced multistage user-based collaborative filtering through non-linear similarity for recommendation systems," *Expert Syst. Appl.*, vol. 161, Dec. 2020, Art. no. 113724.

[45] M. Jalili, S. Ahmadian, M. Izadi, P. Moradi, and M. Salehi, "Evaluating collaborative filtering recommender algorithms: A survey," *IEEE Access*, vol. 6, pp. 74003–74024, 2018.

[46] M. Ranjbar, P. Moradi, M. Azami, and M. Jalili, "An imputation-based matrix factorization method for improving accuracy of collaborative filtering systems," *Eng. Appl. Artif. Intell.*, vol. 46, pp. 58–66, Nov. 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197615001888

[47] T. V. R. Himabindu, V. Padmanabhan, and A. K. Pujari, "Conformal matrix factorization based recommender system," *Inf. Sci.*, vol. 467, pp. 685–707, Oct. 2018.

[48] M.-H. Park, J.-H. Hong, and S.-B. Cho, "Location-based recommendation system using Bayesian user's preference model in mobile devices," in *Ubiquitous Intelligence and Computing*, J. Indulska, J. Ma, L. T. Yang, T. Ungerer, and J. Cao, Eds. Berlin, Germany: Springer, 2007, pp. 1130–1139.

[49] L. Terán and A. Meier, "A fuzzy recommender system for eelections," in *Electronic Government and the Information Systems Perspective*, K. N. Andersen, E. Francesconi, Å. Grönlund, and T. M. van Engers, Eds. Berlin, Germany: Springer, 2010, pp. 62–76.

[50] C. Porcel, A. G. López-Herrera, and E. Herrera-Viedma, "A recommender system for research resources based on fuzzy linguistic modeling," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5173–5183, Apr. 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417408003126

[51] S. Rendle, W. Krichene, L. Zhang, and J. Anderson, "Neural collaborative filtering vs. matrix factorization revisited," in *Proc. 14th ACM Conf. Recommender Syst. (RecSys)*. New York, NY, USA: Association for Computing Machinery, 2020, pp. 240–248.

[52] M. Singh, "Scalability and sparsity issues in recommender datasets: A survey," *Knowl. Inf. Syst.*, vol. 62, no. 1, pp. 1–43, Jan. 2020.

[53] Y. Koren and R. Bell, "Advances in collaborative filtering," *Recommender Systems Handbook*. Boston, MA, USA: Springer, 2015, pp. 77–118, doi: 10.1007/978-1-4899-7637-6_3.

[54] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 2065–2073, Mar. 2014.

[55] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997, doi: 10.1145/245108.245124.

[56] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*. Berlin, Germany: Springer, 2007, pp. 325–341, doi: 10.1007/978-3-540-72079-9_10.

[57] X. Bai, M. Wang, I. Lee, Z. Yang, X. Kong, and F. Xia, "Scientific paper recommendation: A survey," *IEEE Access*, vol. 7, pp. 9324–9339, 2019.

[58] P. B. Thorat, R. M. Goudar, and S. Barve, "Survey on collaborative filtering, content-based filtering and hybrid recommendation system," *Int. J. Comput. Appl.*, vol. 110, no. 4, pp. 31–36, Jan. 2015.

[59] H. Ko, S. Lee, Y. Park, and A. Choi, "A survey of recommendation systems: Recommendation models, techniques, and application fields," *Electronics*, vol. 11, no. 1, p. 141, Jan. 2022. [Online]. Available: https://www.mdpi.com/2079-9292/11/1/141

[60] J. Han, M. Kamber, and J. Pei, "Mining frequent patterns, associations, and correlations: Basic concepts and methods," in *Data Mining* (The Morgan Kaufmann Series in Data Management Systems), 3rd ed., J. Han, M. Kamber, and J. Pei, Eds. Boston, MA, USA: Morgan Kaufmann, 2012, pp. 243–278, ch. 6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012381479100006X

[61] U. Fayyad, "Knowledge discovery in databases: An overview," in *Relational Data Mining*. Berlin, Germany: Springer, 2001, pp. 28–47, doi: 10.1007/978-3-662-04599-2_2.

[62] J. Hipp, U. Güntzer, and G. Nakhaeizadeh, "Algorithms for association rule mining—A general survey and comparison," *ACM SIGKDD Explorations Newslett.*, vol. 2, no. 1, pp. 58–64, Jun. 2000, doi: 10.1145/360402.360421.

[63] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. 20th Int. Conf. Very Large Data Bases (VLDB)*. San Francisco, CA, USA: Morgan Kaufmann, 1994, pp. 487–499.

[64] A. Bansal, A. Khare, and R. Moriwal, "ECLAT algorithm for frequent item set generation with association rule mining algorithm," *Int. J. Sci. Res. Comput. Sci., Eng. Inf. Technol.*, vol. 6, pp. 306–309, Apr. 2020.

[65] P. Singh, S. Singh, P. K. Mishra, and R. Garg, "RDD-Eclat: Approaches to parallelize eclat algorithm on spark RDD framework (extended version)," 2021, *arXiv:2110.12012*.

[66] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 1–12, May 2000, doi: 10.1145/335191.335372.

[67] V. Robu and V. D. dos Santos, "Mining frequent patterns in data using Apriori and Eclat: A comparison of the algorithm performance and association rule generation," in *Proc. 6th Int. Conf. Syst. Informat. (ICSAI)*, Nov. 2019, pp. 1478–1481.

[68] K. Garg and D. Kumar, "Comparing the performance of frequent pattern mining algorithms," *Int. J. Comput. Appl.*, vol. 69, no. 25, pp. 21–28, May 2013.

[69] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications*. Philadelphia, PA, USA: SIAM, 2020.

[70] G. Gan, C. Ma, and J. Wu, *Data Clustering: Theory, Algorithms, and Applications* (ASA-SIAM Series on Statistics and Applied Probability). Philadelphia, PA, USA: SIAM, 2007. [Online]. Available: https://epubs.siam.org/doi/10.1137/1.9780898718348, doi: 10.1137/1.9780898718348.

[71] C.-S. Chang, W. Liao, Y.-S. Chen, and L.-H. Liou, "A mathematical theory for clustering in metric spaces," *IEEE Trans. Netw. Sci. Eng.*, vol. 3, no. 1, pp. 2–16, Jan. 2016.

[72] W. A. Kosters, W. Pijls, and V. Popova, "Complexity analysis of depth first and FP-growth implementations of apriori," in *Machine Learning and Data Mining in Pattern Recognition*, P. Perner and A. Rosenfeld, Eds. Berlin, Germany: Springer, 2003, pp. 284–292.

[73] S. N. Banerjee, "The formula ${}^{n}C_{r}$ revisited," *J. Math. Sci. Comput. Math.*, vol. 1, pp. 75–78, Nov. 2019.

[74] I. Cantador, P. Brusilovsky, and T. Kuflik, "HetRec'11: Proceedings of the 2nd international workshop on information heterogeneity and fusion in recommender systems," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, Chicago, IL, USA. New York, NY, USA: ACM, 2011. [Online]. Available: https://dl.acm.org/doi/10.1145/2039320

[75] A. Kosir, A. Odi, M. Kunaver, M. Tkalcic, and J. Tasic, "Database for contextual personalization," *English Ed.*, vol. 78, pp. 270–274, Jan. 2011.

[76] T. Silveira, M. Zhang, X. Lin, Y. Liu, and S. Ma, "How good your recommender system is? A survey on evaluations in recommendation," *Int. J. Mach. Learn. Cybern.*, vol. 10, no. 5, pp. 813–831, May 2019.

[77] A. Salah, Q.-T. Truong, and H. W. Lauw, "Cornac: A comparative framework for multimodal recommender systems," *J. Mach. Learn. Res.*, vol. 21, no. 95, pp. 1–5, 2020.

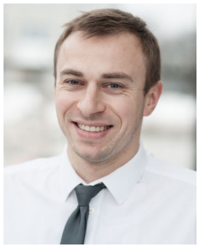
[78] M. Grzegorowski, "Scaling of complex calculations over big data-sets," in *Proc. 10th Int. Conf. Act. Media Technol. (AMT)*, in Lecture Notes in Computer Science, vol. 8610, D. Slezak, G. Schaefer, S. T. Vuong, and Y. Kim, Eds., Warsaw, Poland. Cham, Switzerland: Springer, Aug. 2014, pp. 73–84, doi: 10.1007/978-3-319-09912-5_7.

[79] M. Grzegorowski, "Selected aspects of interactive feature extraction," in *Transactions on Rough Sets XXIII*, vol. 23. Berlin, Germany: Springer, pp. 121–287, 2022, doi: 10.1007/978-3-662-66544-2_8.

[80] M. Grzegorowski, "Massively parallel feature extraction framework application in predicting dangerous seismic events," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Sep. 2016, pp. 225–229.

[81] E. Zdravevski, P. Lameski, R. Mingov, A. Kulakov, and D. Gjorgjevikj, "Robust histogram-based feature engineering of time series data," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, vol. 5, Sep. 2015, pp. 381–388. [Online]. Available: https://ieeexplore.ieee.org/document/7321469, doi: 10.15439/2015F420.
[82] M. Grzegorowski and S. Stawicki, "Window-based feature extraction framework for multi-sensor data: A posture recognition case study," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, in Annals of Computer Science and Information Systems, vol. 5, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., Lódz, Poland, Sep. 2015, pp. 397–405.
[83] E. A. Callejas, J. A. Cerrada, C. Cerrada, and F. J. Cabrerizo, "Group decision making based on a framework of granular computing for multi-criteria and linguistic contexts," *IEEE Access*, vol. 7, pp. 54670–54681, 2019.
[84] H. Naghavipour, M. Y. I. B. Idris, T. K. Soon, R. B. Salleh, and A. Gani, "Hybrid metaheuristics using rough sets for QoS-aware service composition," *IEEE Access*, vol. 10, pp. 112609–112628, 2022.
[85] N. Rehman, A. Ali, M. I. Ali, and C. Park, "SDMGRS: Soft dominance based multi granulation rough sets and their applications in conflict analysis problems," *IEEE Access*, vol. 6, pp. 31399–31416, 2018.
[86] M. Grzegorowski, A. Janusz, J. Litwin, and L. Marcinowski, "Data-driven resilient supply management supported by demand forecasting," in *Proc. 14th Asian Conf. Recent Challenges Intell. Inf. Database Syst. (ACIIDS)*, in Communications in Computer and Information Science, vol. 1716, E. Szczerbicki, K. Wojtkiewicz, S. V. Nguyen, M. Pietranik, and M. Krótkiewicz, Eds., Ho Chi Minh City, Vietnam. Singapore: Springer, Nov. 2022, pp. 122–134.

EYAD KANNOUT received the B.S. degree in informatics engineering from the University of Aleppo, Syria, in 2012, and the M.S. degree in computer science from the Warsaw University of Technology, Poland, in 2016. He is currently pursuing the Ph.D. degree with the Institute of Informatics, University of Warsaw, wherein he is also delivering the laboratories of machine learning course for many years. His research interests include recommendation systems and ML/AI. In addition to his academic career, he held several software engineering positions in the financial sector.



MAREK GRZEGOROWSKI received the Ph.D. degree in computer science, in 2021. For many years, he associated with the Institute of Informatics, University of Warsaw. An Active and Experienced Researcher in the fields of science related to data exploration, machine learning, artificial intelligence, and recommender systems. He has authored a number of scientific articles. In his career, conducted several research and development projects related to the application of ML/AI in academic and industry collaboration.

MICHAŁ GRODZKI received the bachelor's degree in ordinary differential equations, in 2020. He is currently pursuing the master's degree with the Faculty of Mathematics, Informatics and Mechanics, Warsaw University. His research interests include machine learning and recommender systems research.

HUNG SON NGUYEN received the Ph.D. and D.Sci. (Habilitation) degrees, in 1997 and 2008, respectively. He is currently a Professor with the University of Warsaw. On these topics, he has published more than 150 research papers in edited books and international journals and conferences. His main research interests include fundamentals and applications of rough set theory, data mining, text mining, bioinformatics, intelligent multiagent systems, soft computing, and pattern recognition. He was involved in numerous research and commercial projects, including dialog-based search engine, fraud detection, logistic, semantic search engine, intelligent decision support system for firefighting in Poland, RID—development of innovative transport system, recommendation systems, and intelligent chat assistant. He is a fellow of the International Rough Set Society and a member of the Editorial Board of international journals, including, *Transactions on Rough Sets*, *ERCIM News*, and *Computational Intelligence*. He has served as the Manager Editor for Fundamenta Informaticae (until 2017), Data Mining and Knowledge Discovery (2005–2008); the Program Co-Chair for RSCTC'06, RSKT2012, and IJCRS2018; and a PC Member for various other conferences, including PKDD, PAKDD, AAMAS, RSCTC, RSFDGrC, and RSKT.