Ensemble Clustering Approaches Applied in Group-based Collaborative Filtering Supported by Multiple Users' Feedback

Arthur F. da Costa[1]*, Marcelo G. Manzato[1] and Ricardo J. G. B. Campello[2,1]

[1] University of São Paulo, Brazil
{fortes, mmanzato, campello}@icmc.usp.br
[2] James Cook University, Queensland, Australia.
ricardo.campello@jcu.edu.au

Abstract. In this article, we extend our previous work based on group collaborative filtering to improve the quality of groups generated through clustering algorithms with different types of feedback. On the Web, users can interact with content in different ways, such as clicking, commenting or rating and recommender systems should be able to process and use all available information. A pre-processing step using ensemble clustering can be used to combine all this information to create a recommender with better accuracy. In this work, we propose the use of two ensemble clustering approaches to consider different types of feedback and to improve the quality of the recommendations. Experimental results on two different datasets demonstrate that the recommendation accuracy is significantly improved with our approaches when compared to well-known recommender algorithms and with our previous work.

Categories and Subject Descriptors: H.2 [**Database Management**]: Database Applications; H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval

Keywords: Data clustering, Ensemble, Feedback, Recommender systems

1. INTRODUCTION

According to [Gantz and Reinsel 2012], from 2005 to 2020, the information in the digital universe will grow by a factor of 300, from 130 exabytes to 40,000 exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes for every man, woman, and child in 2020). As there is too much information to process and to choose from, it is simply not possible/virtually impossible to grasp even a small percentage of it in a single lifetime. The expression *Information Overload* was introduced to describe the sensation of fatigue and distress that follows the cognitive surplus required to handle the volume of information we have to deal with everyday [Aggarwal 2016].

Recommender Systems (RS) have emerged in response to the information overload problem by learning about users' interests from their past action (ratings, votes, ranked lists, mouse clicks, browse history, product purchases, etc.) and suggesting products that are likely to fit their needs [Aggarwal 2016; Bobadilla et al. 2013]. Collaborative Filtering is one of the most popular and accurate methods used in Recommender Systems. In its simple form it has some limitations such as sparsity, overfitting and cold start [Aggarwal 2016]. Therefore, one solution is the use of hybrid systems, which combine collaborative filtering with additional information describing the contents of the items.

*Corresponding author.

The continuous increase of information, feedback paradigms and content demands some requirements for recommender systems. First is the scalability, that is its ability to generate recommendations quickly using the user-item rating matrix [Bobadilla et al. 2013; Aggarwal 2016], which are of huge dimensionality. Second is to find good items and to improve the quality of the recommendation for a user [Aggarwal 2016]. These two properties are in conflict, since the less time an algorithm spends filtering items, the more scalable it will be, but producing a worse item prediction. Furthermore, these systems typically explore only one type of feedback, discarding possible available knowledge found in other types of feedback. The literature reports a shortage of techniques which integrate different types of user feedback into a generic model [Aggarwal 2016; Bobadilla et al. 2013]. In addition, if different types of feedback are considered in these techniques, this task can increase the computational cost and cause problems in the final result.

Another problem is, once the final recommendation is generated based on the preferences of undistinguished users, that is, the recommender algorithm considers the interactions of all users of the dataset, the system may recommend items for users who have no interest in that selected content (e.g. in a music domain, the system could recommend a pop song for a user that likes metal). In addition, depending on the size of the dataset, the computational cost is very high in order to process all the information [Aggarwal 2016]. A possible solution to these problems would be constructing groups of users with similar interests prior to the recommendation, similarly to when recommending a content to a friend. For the person who receives the recommendation, it works as a filter or a particular view of a universe of possibilities usually inaccessible. It is also possible to make recommendations based on the opinions of others. As for instance, someone who is not an admirer of the jazz genre may be recommended based on what her/his friends enjoy, except this style which is unattractive for him.

In this context, we proposed in a previous work a technique to generate more accurate recommendations using groups of users with similar preferences, called Group-based Collaborative Filtering [da Costa et al. 2016]. We combined a variety of users' feedback types to obtain richer information about their interests. For instance, we captured and combined the ratings and the history of the users to understand and build their profiles based on their behavior in the system. Based on their profiles, we divided these users in groups of preferences considering their tastes. The recommendation list for each user of a particular group is generated using a traditional collaborative recommender algorithm. However, this process was done through a combination of all types of feedback, which can result in the loss of semantic information and the actual distribution of data in each type of feedback, since each type of feedback has its own characteristics (e.g. ratings are decimal numbers and explicitly express the taste of users, while the history are boolean numbers and represent whether or not users interacted with an item).

In this article, we extend our previous work to improve the quality of groups generated through ensemble clustering techniques, in order to maintain the characteristics of the data in the clustering process. We propose the use of two ensemble clustering techniques to combine the final results of each clustering algorithm based on each type of feedback. We evaluated our proposal by comparison with two well-known collaborative recommenders (User KNN and BPR MF), and then compared different types of recommendations generated from our previous approach to demonstrate the proposal's generality and efficiency. The experiments were executed with two real datasets from different domains: the first is MovieLens, which contains data from a movies reviews system; and the second is Last FM, a music website. Our study shows that the proposed group-based approach is able to provide better accuracy than individual recommenders and our previous work.

This article is structured as follows: Section 2 addresses the related work; Section 3 describes techniques which are explored in this work; Section 4 discusses our previous work related with this article; Section 5 presents the proposal in details; Section 6 reports the evaluation executed in the system; finally, Section 7 presents the final remarks and future works.

2. RELATED WORK

In this section, we review some work related to our proposal, as well as the main advantages of our work when compared to these related models. First, we depict approaches related to different types of feedback in recommender systems, and then, we provide a review of data clustering approaches in recommender systems.

2.1 Multiple Feedback Based Approaches

With the increasing number of feedback between users and content, several studies have emerged to work with the integration of these interactions, so that more information about users preferences are gathered by the systems. The recommendation systems can be extended in various ways in order to improve the understanding of users and items, including, for example, new types of interaction in the recommendation process and their combination [Aggarwal 2016; Bobadilla et al. 2013].

The SVD++ recommender algorithm proposed in [Koren 2010] uses explicit (ratings) and implicit (viewing history) information from users in a factorization model. Another factorization model, called Factorization Machines (FM) [Rendle 2012], can consider many types of information regarding users, items and/or their interactions. These techniques have the drawback that they process only certain types of interactions, with little capability of extension to other different types or they do not consider the semantics of the data and the context of each type of feedback. Costa and Manzato [da Costa Fortes and Manzato 2014] developed an ensemble recommender technique, called Ensemble BPR Learning, to unify different types of feedback from users, processed by different recommender techniques. While this model is extensible for any types of user's interaction, its learning phase has high computational cost, since it depends on the execution of several recommendation techniques beforehand and a post-processing step for learning weights to combine each feedback type ranking.

In a recent work, Peska [Peska 2016] proposed a recommender framework that aims to bridge the data sparsity problem and the lack of relevant feedback by modeling and utilizing enhanced sources of information, foremost implicit user feedback features. More specifically, his work focuses on the question how to define and collect multiple user feedback in scenarios, where we cannot invasively ask users to provide it. His results were able to improve quality of recommendations over both binary feedback baseline and uncontextualized feedback in terms of the evaluation metrics used in his study.

However, in these works the authors did not investigate the influence and usage of each contextual feature separately or the possibility to combine purchase probabilities coming from different learning methods or recommender algorithms.

2.2 Clustering Based Approaches

In this work, we have used data mining techniques to cluster users with similar preferences to generate recommendations based on their group. Several researchers have proposed recommender systems for on-line personalization through data mining to provide recommendation services. This kind of recommendation system is used to predict the user navigation behavior and their preferences using web log data.

Kim et al. [Kim et al. 2002] developed a recommender algorithm that uses the k-Means clustering to reduce the dimensionality of the data during the recommendation process. In their work, they used a graph approach to choose the best cluster with respect to a given test customer in selecting the neighbors with higher similarities as well as lower similarities. Their approach allows to explore the transitivity of similarity in a pre-processing step and considers the attributes of each item. The work developed by Wen and Zhou [Wen and Zhou 2012] presents an improved collaborative filtering recommendation algorithm based on dynamic item clustering method. A similarity threshold limits the similarity between clusters. By calculating the similarity between the current item and the cluster center, it chooses the greatest similitude cluster, and then it finds the target items' nearest neighbors. Li et al. [Li and He 2013] propose a collaborative filtering recommender, which uses clustering methods based on users and items. Their approach consists of classifying and ranking the users in multiple item clusters by computing their rating qualities based on the previous rating records. In turn, the items are recommended for target users according to their similar users with high-ranks in different item categories. Experiments on public datasets have demonstrated the effectiveness of their algorithm.

In recent studies, Gupta and Patil [Gupta and Patil 2015] developed an algorithm using a hierarchical clustering algorithm along with a voting scheme to recommend the rating of a particular user with respect to a particular item on movies domain. In their approach, the users with their features are taken and are clustered into different groups using hierarchical clustering. Then, given a pair user and item, the rating prediction is done by mapping a user into a particular group she/he belongs to and then applying voting scheme for all users present in that cluster for a specific item. Katarya and Verma [Katarya and Verma 2016] also present a recommender system based on movies domain through data clustering and computational intelligence. In their research article, a novel recommender system makes use of k-means clustering by adopting cuckoo search optimization algorithm to find the best results based on the most suitable weight among all possible ones applied on the Movielens dataset. Their approach delivers better results than the baselines using mean absolute error, standard deviation, and root mean square error.

Our work is different from related work because we analyze various types of users' feedback on a particular item, in order to create groups of users with similar preferences and thus, a more accurate user's profile. In our previous work [da Costa et al. 2016], we proposed a pre-processing step, which was responsible for generating a single distance matrix weighted from different types of feedback. The advantages of this approach are the ease of extending the template for insertion of other types of feedback and the reduced time and computational processing, considering that the sets of data would be reduced to a single cluster before computing the recommendation. However, in this process we may end up missing important features, such as semantics and data distribution, in the clustering process. In the present work, our model is extended to support these characteristics, processing each type of feedback individually in the clustering algorithm and making only the combination of the final clustering result.

3. RELATED MODELS OVERVIEW

This study involves the pre-processing of data by means of data clustering techniques and the recommendation of items through collaborative filtering algorithms. The following sections present the main concepts covered in this article.

3.1 Notation

In this article, we use a consistent mathematical notation for referencing elements of the recommender system works. The feedback matrix is denoted by $\mathbf{R}$, with r_{ui} being the explicit or implicit interaction that user u provided for item i. In the first case, it is an integer provided by the user indicating how much she/he liked the content; in the second, it is just a boolean indicating whether the user consumed or visited the content or not. The set of pairs (u, i) for which known interaction in $\mathbf{R}$ are represented by the set $K = \{(u, i)|r_{ui} \text{ is known}\}$. $N(u)$ is the set of items for which user u provided a feedback, $\bar{N}(u)$ to indicate the set of items that are unknown to user u and K. Finally, the prediction of the recommender algorithm for the pair (u, i) is represented by $\hat{r}_{ui}$, which is a floating point value guessed by the recommender algorithm.

3.2 Clustering Approaches

Clustering is the assignment of objects into clusters, so that objects from the same cluster are more similar to each other than objects from different clusters [Estivill-Castro 2002]. Often similarity is assessed according to a distance measure, e.g. Euclidean, Cosine, Correlation, Jaccard, etc. Clustering is a common technique for statistical data analysis, which is used in many fields, including recommender systems. In the following subsections we present the main clustering concepts related to this article.

3.2.1 *K-Medoids.* This algorithm is a partition-based clustering algorithm based on k-means. It attempts to minimize the distance between points labeled to be in a cluster and a point designated as the center of that cluster [Park and Jun 2006]. The k-medoids algorithm differs from k-means because it chooses datapoints as centers (medoids or exemplars) and works with an arbitrary matrix of distances between these datapoints, besides minimizing a sum of pairwise dissimilarities instead of a sum of squared Euclidean distances [Park and Jun 2006]. A medoid can be defined as the object of a cluster whose average dissimilarity to all the objects in the cluster is minimal, i.e., it is the most centrally located point in the cluster. The implementation of the k-medoids used in this article is explained as follows:

K-Medoid Clustering Algorithm

```
(1) Initialization: randomly select (without replacement) $k$ of the $n$ data points as the medoids.
(2) Associate each data point to the closest medoid.
    (Using a dissimilarity measure like cosine, Pearson, etc.)
(3) For each medoid $m$
    —For each non-medoid data point $o$
      —Swap $m$ and o and compute the total cost of the configuration
(4) Select the configuration with the lowest cost.
(5) Repeat steps 2 to 4 until there is no change in the medoid.
```

The advantage of this k-medoids implementation is that large datasets can be efficiently classified and its convergence is proved regardless of the dissimilarity measure. Furthermore, it is reliable in theory, simpler and faster than k-means, since it does not calculate the distance between data points. [Park and Jun 2006]. In this study, we will use k-medoids to generate the most similar groups of users, in order to reduce the dimensionality and sparsity of the data, since this process will cause the recommendation to be computed based only on the group's preferences of each user.

3.2.2 *Ensemble Clustering.* Strehl and Ghosh [Strehl and Ghosh 2003] introduce the problem of ensemble multiple clusters of a set of objects into a single consolidated group without accessing the features or algorithms that determined these partitions. This approach, also called consensus clustering or aggregation of clustering, refers to the situation in which a number of different clusterings have been obtained for a particular dataset and it is desired to find a single clustering which is a better fit in some sense than the existing clusterings [Punera and Ghosh 2008].

In Strehl and Ghosh work [Strehl and Ghosh 2003], they discuss three approaches towards solving this problem to obtain high quality consensus functions, which have low computational costs. This characteristic makes them feasible to evaluate each of the techniques discussed below and find the best results by comparing the solution against the objective function. Their proposed hard ensemble clustering methods are Cluster-based similarity partitioning algorithm (CSPA), Hyper-graph partitioning algorithm (HGPA) and Meta-clustering algorithm (MCLA). In CSPA the similarity between two datapoints is defined to be directly proportional to the number of constituent clusterings of the ensemble in which they are clustered together [Punera and Ghosh 2008]. The main idea is that the more related two data-points are the higher is the chance of belonging to the same cluster. This similarity graph between data-points is partitioned using METIS[⋆] to obtain the desired number of clusters. CSPA is the simplest heuristic, but its computational and storage complexity are both quadratic in n [Punera and Ghosh 2008].

In turn, according to Punera and Gosh [Punera and Ghosh 2008], the HGPA algorithm takes a very different approach to find the consensus clustering than the previous approach, in which partitioning the hypergraph by cutting a minimal number of hyperedges was formulated. Finally, the MCLA algorithm is based on clustering clusters, where each cluster is also represented as a hyperedge. The algorithm groups and collapses related hyperedges into k clusters; and then assigns each data point to the collapsed hyperedge in which it participates most strongly [Punera and Ghosh 2008].

In this article we evaluate the performance of some of these ensembles techniques on more complex real-life datasets in the recommender systems' context. One advantage of these approaches is that they enable us to conveniently model final clusters of different sizes via priors in the mixture model. Graph partitioning methods tend to yield roughly balanced clusters. However, this is a disadvantage in situations where the data distribution is not uniform.

3.3 Recommendation Algorithms

We evaluate our proposal with two recommender algorithms: a neighborhood-based [Koren 2010; Aggarwal 2016] and a matrix factorization [Rendle et al. 2012] approach. The following subsections detail each algorithm.

3.3.1 *BPRMF.* This approach [Rendle et al. 2012] consists of providing personalized ranking of items to a user according only to implicit feedback (e.g. navigation, clicks, etc.). An important characteristic of this type of feedback is that we only know the positive observations; the non-observed user-item pairs can be either an actual negative feedback or simply the fact that the user does not know about the item's existence. The authors have proposed a generic method for learning models for personalized ranking, where instead of training the model using only the user-item pairs, they also consider the relative order between a pair of items, according to the user's preferences [Rendle et al. 2012]. It is inferred that if an item i has been viewed by user u and j has not ($i \in N(u)$ and $j \in \bar{N}(u)$), then $i >_u j$, which means that she/he prefers i over j. Each user u is associated with a user-factors vector $p_u \in \mathbb{R}^f$, and each item i with an item-factors vector $q_i \in \mathbb{R}^f$, where f is the number of factors in each vector.

It is important to mention that when i and j are unknown to the user, or equivalently, both are known, then it is impossible to infer any conclusion about their relative importance to the user. To estimate whether a user prefers an item over another, Rendle et al. proposed a Bayesian analysis using the likelihood function for $p(i >_u j|\Theta)$ and the prior probability for the model parameter $p(\Theta)$. The final optimization criterion, BPR-Opt, is defined as:

$$\text{BPR-Opt} := \sum_{(u,i,j)\in D_K} \ln \sigma(\hat{s}_{uij}) - \Lambda_\Theta ||\Theta||^2 \quad , \tag{1}$$

where $\hat{s}_{uij} := \hat{r}_{ui} - \hat{r}_{uj}$ and $D_K = \{(u,i,j)|i \in N(u) \ \& \ j \in \bar{N}(u)\}$. The symbol Θ represents the parameters of the model, Λ_Θ is a regularization constant, and σ is the logistic function, defined as: $\sigma(x) = 1/(1+e^{-x})$.

For learning the model, the authors also proposed a variation of the stochastic gradient descent technique, denominated LearnBPR, which randomly samples from D_K to adjust Θ.

[⋆]http://glaros.dtc.umn.edu/gkhome/metis/hmetis/overview

3.3.2 *User KNN.* This recommendation algorithm is the well-known User KNN, whose details can be found in [Koren 2010; Aggarwal 2016]. We adopted this algorithm because of its well-acceptance, and because it can be intuitively extended to include other information. The main goal of the algorithm is to find similar users and predict the best items for them based on their similar items.

In this way, a score is predicted for a unknown user-item pair $\hat{r}_{ui}$ considering the interaction that other users with similar preferences to u have assigned to item i. To find similar users, a measure of similarity p_{uv} is employed between their vectors. The similarity measure may be based on several similarity measures, such as Pearson correlation coefficient or cosine similarity. The final similarity measure is a retracted coefficient, s_{uv}:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv}, \tag{2}$$

where n_{uv} is the number of items that users u and v have in common, and λ_1 is a regularization constant, set as 100 according to suggestions found in the literature [Koren 2010].

Using the similarity values obtained, the algorithm identifies the k most similar users of u who evaluated item i, denoted as $S^k_u(i; v)$, and performs a score prediction based on the interactions of the k similar users weighted by their similarity towards u [Koren 2010]. Then the final score is predicted through the Equation (3).

$$\hat{r}_{ui} = \frac{\sum_{v \in S^k(i;u)} s_{uv}}{k} \tag{3}$$

4. GROUP-BASED COLLABORATIVE FILTERING APPROACH

In our previous work [da Costa et al. 2016], we proposed an approach capable of generating recommendations based on users groups' preferences. This approach consisted of a pre-processing module, responsible for combining users into groups according to different types of feedback. In this way, the combination of tags assignments and user history during navigation, for example, could be made to improve the quality of the groups, since they can better represent the behavior of each user. The recommendation list for each user of a particular group is generated using a well-known recommender algorithm based on collaborative filtering.

In a first step, multiple users' feedbacks are captured and used to generate user vs. item matrices, where each cell in these matrices is a value containing the relevance of each feedback type. These matrices are then used to compute the distance of users using some dissimilarity measure. After this step, the algorithm generates a single distance matrix resulting from the combination of the others. Then, this matrix is used by the clustering module to generate users' groups. Each of these clusters corresponds to users who have similar interests on particular subjects. Finally, based on these groups, particular recommendations are computed to each user in the dataset, through well-known recommender algorithms based on collaborative filtering, using the navigation history (implicit feedback) of each group. The browsing history of each group is built using all kinds of interactions considered by the algorithm, making explicit interactions in binary form and removing duplicate entries. There are three phases in our approach: data representation; data clustering; and recommendation. The following subsections detail each of them.

4.1 Data representation

The algorithm inputs are represented by user $\times$ item interactions matrices, so if the system's data are compared by two types of feedback (e.g. ratings and tags assignments), we will have two input matrices. Each cell in the matrix represents the interactions made by users on items. Different methods can be used to represent interactions. Discrete values (such as 1, 2, 3, 4, 5) can be used to represent degrees of user's preferences towards the items; numerical values can be used to characterize the amount of times a user accessed an item; boolean values (such as 0 and 1) to represent whether a user assigned or not a tag on an item. In this way, each cell of each matrix gets its respective type of interaction (explicit or implicit). If a user has not interacted with the corresponding item, its value in the matrix is 0, otherwise it will be specific for each type of interaction. For example, if the user explicitly rated an item, this value will be the provided rating in an explicit feedback matrix; if she/he has only viewed the item, this value will be 1 in an implicit feedback matrix.

4.2 Data Clustering

Data Clustering is the process of grouping a set of objects into clusters so that objects within a cluster are similar to each other but are dissimilar to objects in other clusters [Han et al. 2001]. The similarity between a user and other users is acquired based on their interactions. In this work were used Cosine angle and Pearson correlation to calculate how two users are alike, considering all the interactions made by users on all items in the database. These metrics were chosen because: i) they discard the matrix cells that have no interaction, and ii) they are the most commonly used metrics in the area of recommender systems [Bobadilla et al. 2013; Aggarwal 2016]. This is particularly useful because we can not assume that users are similar based on the fact they have not interacted with certain items. For each feedback, these metrics were used to generate a new matrix of $M \times M$ users, which represents the dissimilarity among users. To combine the distances of each type of interaction in a single distance matrix, a weighted average of the values was computed, as shown in Equation (4).

$$d^{final}_{(u,v)} = \frac{1}{N_f} \sum_{n=1}^{|N_f|} \frac{1}{\alpha_n} d_n \tag{4}$$

where N_f is the number of types of feedback, d_n is the distance calculated based on each type of interaction and α_* are variables used to weight each interaction type, which are defined as:

$$\alpha = \frac{N_{uv}(N_u + N_v)}{(N_u N_v)}. \tag{5}$$

In the equation above, N_u and N_v denote the number of interactions made by users u and v, respectively, and N_{uv} denotes the number of interactions in common of these users.

After computing the distance matrix, the approach used k-medoids, presented in Section 3.2.1, to generate groups. In this way, in this work k data objects are selected randomly as medoids to represent k clusters and all remaining data objects are placed in a cluster having a medoid nearest or most similar to that data object. In the next step, a new medoid is determined which can represent the cluster in a better way and the entire process is repeated. Again all data objects are bound to the clusters based on the new medoids. In each iteration, medoids change their location step by step; in other words, medoids move in each iteration. This process is continued until all medoids stop moving over each iteration. As a result, k clusters are found representing a set of n data objects.

4.3 Recommendation

In this step, well-known CF-based algorithms were used to process the interactions of each cluster and generate a list of recommended items for each user in that cluster. At this stage, each user receives personalized recommendations based on her/his behavior and her/his neighbors behavior. The algorithm is responsible for assembling a matrix that contains all users and items of a given group k and individual interactions of each user to predict items that she/he can enjoy, both highlighting the items she/he visited as the ones she/he has not. Finally, the individual recommendations are concatenated in a single ranking with all users, which contains pairs (u, i) sorted by scores generated by the recommenders. The proposed technique generates four values of k from the training set of samples. In this step, the sample is divided into training and test in order to verify the precision and MAP values generated by this sample; then this procedure is repeated n times, returning the best values for k.

5. PROPOSED APPROACHES

In our previous work, the process was done through a combination of all types of feedback before a clustering step in a merge of distance matrices, which can result in the loss of semantic information and the actual distribution of data in each type of feedback. Each type of feedback has its own characteristics and can be expressed in different ways, for example ratings are decimal numbers and explicitly express the taste of users, while the history are boolean numbers and represent whether or not users interacted with an item. The number of interactions and semantic value attributed by the user in each type of feedback can directly influence the representation of her/ his behavior during the clustering and the recommendation processes and the combination of these interactions prior to the clustering algorithm can generate noise and data distortion, since the distance matrix is made only with heuristic weighting.

In this article we extend the work presented in Section 4, in order to conserve the considered characteristics of each type of feedback during the clustering process. In this way, we propose a generic ensemble clustering to combine multiple feedback types extracted from datasets and we also use ensemble clustering algorithms well-known in the literature [Strehl and Ghosh 2003]. The ensembles are accomplished in a pre-processing module based on clustering approach, which combine the outputs generated by a clustering algorithm using individual users' feedback types. Our proposed approach allows the use of different types of clustering and recommendation algorithms, which makes it generic and extensible. Figure 1 shows a representation of the proposed approach.

In this article, we process each feedback type in a separate way, combining only the final result of the clusterings. In this way, we are able to preserve the characteristics and distribution of the data of each type of feedback in the clustering process. In Step (1), we build a user vs. item representation in the same way as in previous work, where each value in these representations is a score representing each feedback type. These matrices are then used to compute the dissimilarity of users using some distance measures. Then, each distance matrix is processed by a clustering algorithm, which is responsible for generating groups of users based on their preferences.

Then, in Step (2), the results generated by the clustering algorithm for each type of interaction are combined in an ensemble clustering process, in order to improve the quality of the generated groups. In this article, we propose the use of two strategies of ensemble clustering. The first ensemble clustering technique is based on heuristics, which combines individual results of each feedback generated by clustering algorithm using vote strategy, where the most common group label for each user in the results of each approach is considered the real label group of that user. For a more reliable and robust tool, the second technique was based on CSPA, HGPA and MCLA [Strehl and Ghosh 2003], called Consensus Clustering Based Strategy, in which a user's label is defined by well-known clustering ensembles strategy.

Finally, based on the groups defined by ensemble clustering strategies, we compute particular recommendations to each user in the dataset, using all feedback of each group int the Step (3). The main difference of this new approach is the implementation of an ensemble process after the clustering

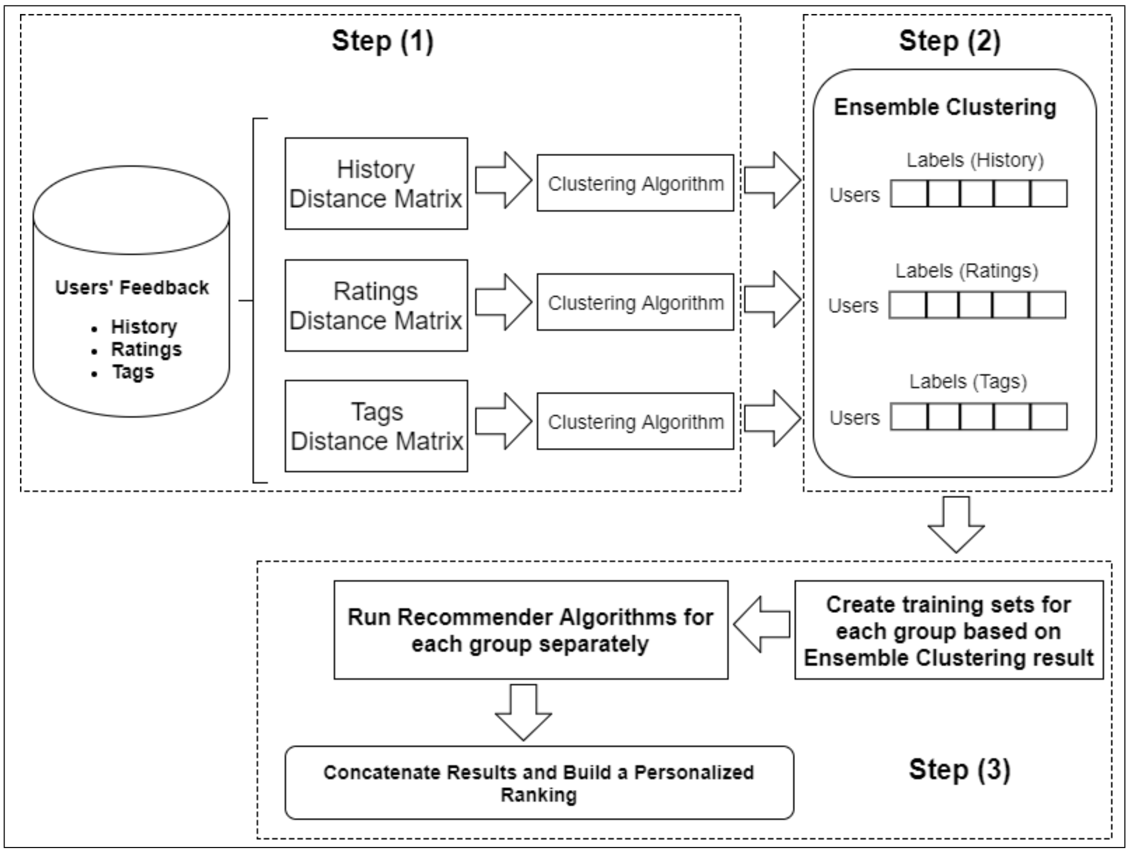

Fig. 1. Schematic visualization of the proposed approach.

step, instead of a weighted combination of the distance matrices of each type of feedback. Thus, the other steps of the approach are the same as those presented in Section 4 and the proposed ensemble strategies are detailed in the following subsections.

5.1 Voting Strategy

In this ensemble technique, the target is to combine multiple clustering solutions or partitions of a set into a single consolidated clustering that maximizes the information shared among all available clustering solutions. We are looking for a final partition $C*$ of a given dataset $\{user_1, ..., user_N\}$ into k labels which optimally represent a given set of M partitions of the different feedback sets. Each of these M partitions is generated by a clustering algorithm for a given type of feedback and is represented by a vector with N positions, where each cell of this vector contains the label on which a particular user u belongs to. In this way, we create an auxiliary matrix $A_{N\times k}$ to count the number of votes of each label for each user taking into account all types of interaction. The element a_{ij} is the degree of membership of $user_u$ to the j-th label of the m-th partition. For every type of feedback M, a element a_{ij} in matrix A is defined by:

$$a_{uj} = a_{uj} + \begin{cases} 1 & \text{if } user \text{ is assigned in cluster } j, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The final partition $C*$ is encoded as a vector with elements c_u, where each cell is the ensemble clustering result of the combination of the M feedback types, defined by:

$$c_{user_u} = \arg\max a_u \tag{7}$$

The main idea of this approach is to keep the most common label assigned by the clustering algorithm to a given user, since the repetition of this assignment in more than one type of feedback provides a greater chance of a given user to be part of a group k. Figure 2 illustrates an example for a given user u, which interact with the system with ratings, tags and history.

Once the clustering algorithm is applied to each type of feedback, our approach evaluates the groups that were assigned to the user i and then assigns him/her to the group in which he/she had the most votes based on all considered feedback (in the case of the example in Figure 2, user u is assigned to the group 0).

5.2 Consensus Clustering Strategy

In this ensemble approach, we aggregate the clustering information, searching for a consensus clustering that is on average the most consistent with the different M partitions in the ensemble, using a strategy proposed by Strehl and Ghosh [Strehl and Ghosh 2003]. For a population of n cells, the similarity between a pair of clusterings, λ^a and λ^b , which contains k^a and k^b clusters respectively, is quantified by the normalized mutual information (NMI), defined as:

$$\phi^{(NMI)}(\lambda^a, \lambda^b) = \frac{\sum_{h=1}^{k^a} \sum_{l=1}^{k^b} n_{h,l} log(\frac{n \cdot n_{h,l}}{n_h^a n_l^b})}{\sqrt{(\sum_{h=1}^{k^a} n_h^a log(\frac{n_h^a}{n}))(\sum_{l=1}^{k^b} n_l^b log(\frac{n_l^b}{n}))}} \quad , \tag{8}$$

where n_h^a and n_l^b denote the numbers of cells in the corresponding clusters, and $n_{h,l}$ stands for the number of cells in their intersection.

For an ensemble of M partitions, $\lambda^1, \ldots , \lambda^M$, the consensus clustering $\lambda^* = \{C_1^*, ..., C_K^*\}$ is defined as the one that maximizes the average NMI with the M partitions in the process. The final result is computed by combining three approximation algorithms, CSPA, HGPA and MCLA, and selecting the one that performs the best [Strehl and Ghosh 2003].Thus, each algorithm is executed n times, and generates an approximate solution for each approach. Finally, one chooses the best result among all approaches.

6. EVALUATION

To evaluate our proposal, we first compare our ensemble techniques against each individual collaborative recommender. This comparison was carried out regarding each feedback type. Then, we compare the different types of ensembles against our previous work [da Costa et al. 2016], in order to

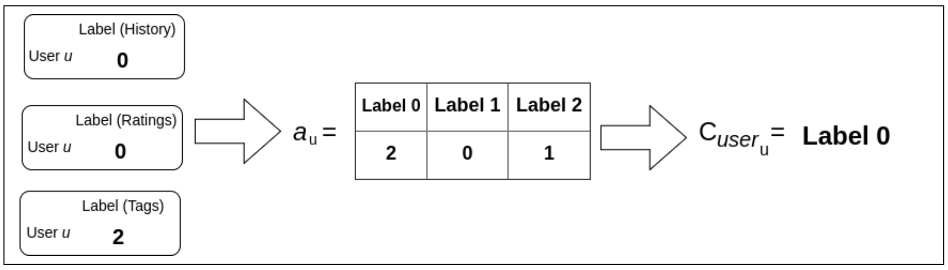

Fig. 2. Example of Ensemble based on Voting Strategy.

demonstrate which one provides the best results. This evaluation was executed for both considered collaborative recommenders, User KNN and BPRMF, the clustering algorithm, K-Medoids, and two datasets, as follows.

6.1 Datasets

The system evaluation was based on two datasets provided by Cantador et al. [Cantador et al. 2011]. Last fm 2k consists of 92,834 user-listened artist relations, 186,479 interaction tags applied by 1,892 users to 17,632 artists. As feedback types, we considered: whether a user tagged an item or not (e.g. a user annotate an item with freely chosen keywords based on her/his taste); and the number of times the user has visited a particular item. MovieLens 2k consists of 10 million ratings, 100,000 interactions tags applied to 10,000 users and 72,000 movies. As explicit information, we used the ratings that users assigned to items to calculate the distance matrix, and as implicit information, we considered whether a user tagged an item or not and the navigation history to compute matrices and recommendations.

6.2 Used Resources

The recommender approach proposed in this article was developed in Python[⋆⋆] version 2.7, with NumPy[⋆⋆⋆] and SciPy[†] libraries, responsible for data optimization and matrix structures. Its source-code is freely available on Github[‡].

The recommender algorithms integrated into the tool belong to MyMediaLite library [Gantner et al. 2011] and Case Recommender [da Costa Fortes and Manzato 2016], which are open-source tools, developed in $C\#$ and Python, with numerous features and algorithms for recommender systems. Among the algorithms implemented are User KNN and BPR MF, both used in this study. For Consensus Clustering, we used an ensemble clustering Python library, called Cluster Ensembles version 1.16 [§].

6.3 Experimental Setup and Evaluation Metric

We adopted the All But One [Breese et al. 1998] protocol for the construction of the ground truth and 10-fold cross-validation. Given the data set, we randomly divided it into the same 10 subsets and for each sample we use $n-1$, these subsets of data for training and the rest for testing. The training set t_r was used to test the proposed technique and in the test set T_e we randomly separated one item for each user to create the truth set H. In this way, the truth set H will have only one item for each user of the original test set and the evaluation of the generated rankings will be made using this set. To assess the outcomes of the systems we use the evaluation metric Mean Average Precision (MAP) [Voorhees and Harman 2005], as follows:

Mean Average Precision computes the precision considering the respective position of items in the ordered list. With this metric, we obtain a single accuracy score value for a set of test users T_e:

$$MAP(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} AveP(R_j, H_j), \tag{9}$$

[⋆⋆]http://www.python.org/

[⋆⋆⋆]http://www.numpy.org/

[†]http://www.scipy.org/

[‡]http://www.github.com/ArthurFortes/CaseRecommender/

[§]https://pypi.python.org/pypi/Cluster_Ensembles/1.16

where R is the set of recommendations that the system computed, given the set of observables O, and the ground truth set H. The average precision (AveP) is given by

$$AveP(R_j, H_j) = \frac{1}{|H_j|} \sum_{r=1}^{|H_j|} [Prec(R_j, r) \times \delta(R_j(r), H_j)], \tag{10}$$

where $Prec(R_j, r)$ is the precision for all recommended items up to rank r and $\delta(R_j(r), H_j) = 1$, if the predicted item at rank r is a relevant item ($R_j(r) \in H_j$) or zero otherwise.

In this work we used MAP@N, where N corresponds to the number of recommendations. We tested for the following values: $1, 3, 5$ and 10 in the ranks returned by the system. For each configuration and measure, the 10-fold values are summarized by using mean and standard deviation. In order to compare the results in statistical form, we apply the two-sided paired t-test with a 95% confidence level [Mitchell 1997].

Regarding the parameters of the algorithms, we defined a set of values which performed better for both datasets, HetRec MovieLens 2k and LastFm. Such definitions were made by cross-validation in the training set and some techniques like "Knee Finding" to choose the best number of clusters [Estivill-Castro 2002]. For the k-Medoids algorithm, we ran experiments for k equals to 3, 5, 7, 10, 30, 50 and 100 neighbors, and chose k equals to 3 as it provided the best results. To execute Consensus Clustering Strategy, we ran experiments for n equals to 2, 5, 10 and 15 times, and chose $n = 5$ as it provided the best results. For the BPRMF algorithm, we ran experiments for the latent factors 10, 40, 70 and 100, and chose the number of 40 latent factors as it presented the best results for this algorithm. For User KNN, we ran experiments for k equals to 30, 50, 80 and 100 neighbors, and chose k equals to 50 as it provided the best results. For User KNN and K-Medoids, we used Cosine Similarity to generate the distance and similarity matrices, respectively, once the results indicate that this similarity is superior than the other measures such as Pearson Correlation and Jaccard measure that we tested. For other parameters, we use default values of each recommender tool and library chosen.

For the final results, once the medoids are chosen randomly in the clustering algorithm, we run ten times each experiment and choose the best value generated. In addition, we restricted the clusters to have no less than 10 users, since we assume that this is the minimum value to generate a good neighborhood, as well as a good recommendation for a given user.

6.4 Results

In our previous work, we compared the Group-based approach using each type of feedback with well-known recommender algorithms, which were described in Section 3.3. Then, we compared our approach using all types of feedback with each recommender algorithm using a training set, which was built based on the concatenation of the training sets of each feedback, demonstrating the accuracy improvement of the proposed approach in relation to baselines.

In this article, the results of the experiments in each of the datasets are discussed in the following subsections, in which we first compare our proposed approach using all interactions to each recommender algorithm executed with each type of feedback and then compare our approach to our previous work using all types of feedback.

6.4.1 *Last Fm 2k.* In this experiment, we considered two different types of implicit feedback available in the LastMF 2k dataset, history and tags. Table I shows the results using the combination of both types of feedback considered in this dataset (history and tags) against the best results in the baselines trained with each feedback separately. Each ensemble clustering and Group-based approach

Table I. Comparison among Ensemble Clustering approaches and recommender algorithms in terms of Map@N (Last Fm 2k).

Algorithms	**Feedback**	**Map@1**	**Map@3**	**Map@5**	**Map@10**
User KNN					
Traditional Recommender	History	0.101986	0.208803	0.271068	0.358561
	Tags	0.040257	0.096081	0.137412	0.205045
Group-Based	History and Tags	0.117552	0.223832	0.278582	0.365002
Voting Strategy	History and Tags	0.124530	0.243156	**0.301127***	0.387546
Consensus Clustering	History and Tags	**0.127750***	**0.244766**	0.298443	**0.390164***
BPRMF					
Traditional Recommender	History	0.052603	0.104133	0.1438539	0.213097
	Tags	0.030595	0.059044	0.088031	0.132581
Group-based	History and Tags	0.071853	0.134971	0.180891	0.256575
Voting Strategy	History and Tags	0.082662	0.151905	**0.205045***	0.273752
Consensus Clustering	History and Tags	**0.085346***	**0.155126***	0.201825	**0.276435***

Bold typeset indicates the best performance. * indicates statistical significance at p <0.01 pairwise compared to the other results.

Table II. Comparison among Ensemble Clustering approaches and recommender algorithms in terms of Map@N (MovieLens 2k).

Algorithms	**Feedback**	**Map@1**	**Map@3**	**Map@5**	**Map@10**
User KNN					
Traditional Recommender	History	0.023300	0.067047	0.100808	0.161184
	Ratings	0.029481	0.070375	0.105087	0.163086
	Tags	0.003804	0.013789	0.021398	0.038040
Group-Based	History, Ratings and Tags	0.026628	0.072753	0.108416	0.172610
Voting Strategy	History, Ratings and Tags	0.029481	**0.075130***	0.109843	0.166428
Consensus Clustering	History, Ratings and Tags	**0.030153***	0.071326	**0.113185***	**0.179365***
BPRMF					
Traditional Recommender	History	0.014741	0.050404	0.092249	0.147883
	Ratings	0.021398	0.061816	0.092724	0.154065
	Tags	0.002377	0.007608	0.013789	0.029481
Group-Based	History, Ratings and Tags	0.026153	0.068473	0.097479	0.150016
Voting Strategy	History, Ratings and Tags	0.026153	**0.069900***	0.097479	0.156916
Consensus Clustering	History, Ratings and Tags	**0.031383***	0.069473	**0.109161***	**0.164051***

Bold typeset indicates the best performance. * indicates statistical significance at p <0.01 pairwise compared to the other results.

result is applied to the chosen recommender algorithms (User KNN and BPRMF). The best results are highlighted in bold.

As it can be seen, the ensemble approaches based on the different types of feedback provided the best results in the vast majority of cases. In both datasets and algorithms, we could achieve a gain by enriching the representations with different types of feedback. The most robust model, Consensus Clustering, provided the best results, giving substantial indication that, for the clustering scenarios addressed, it can find suitable similarities between groups.

6.4.2 *MovieLens 2k.* This dataset contains several implicit and explicit feedback. For this experiment, we used as explicit feedback the ratings and as implicit feedback the history and tags assignments. The same experiments were performed in the MovieLens dataset and the results are presented on Table II.

Again, one can see that the representations that used ensemble clustering strategies provided the best results when compared to traditional approaches. Considering the results in the MovieLens

2k dataset, the same effect happens when we used ensemble clustering approaches compared with traditional recommender and with our previous work. This happens because the descriptive power of the enhanced representations of each type of feedback applied separately in a clustering algorithm is superior to the normal representation, giving better close-related neighbors in each feedback and the data semantics.

6.5 Analysis and Discussion

As detailed in the previous sections, the use of ensemble clustering approaches provides better results than combining distance matrices and processing individual feedback for most cases, which indicates that our proposal has potential to improve results provided by recommender systems. In Tables I and II, we see that by using the history and ratings, we obtain better values of MAP than using tags in each recommender algorithm separately. But, if we combine all feedback types using our proposed clustering ensemble approaches, the results are even better than the previous combination. We can also see in the experiments that the proposed approaches using different types of feedback provide better results than a collaborative filtering based on matrix factorization (BPRMF) and neighborhood (User KNN), demonstrating that by using different types of feedback we can provide better recommendations.

We also notice that the ensemble clustering approaches proposed in this article provided the best results for both recommender algorithms than our previous work. Consensus clustering approach provided the best results for the BPRMF and User KNN algorithms for the two datasets, while Group-based approach provided the worst results in most of the cases when compared to ensemble approaches. However, as it can be seen in our previous work [da Costa et al. 2016], the group-based approach can also overcome all the baselines run separately with each type of feedback.

The Voting Strategy provided results close to the Consensus Clustering and was proposed to have a low computational cost and be easy to implement. In this way, this approach may be an alternative to scenarios with a larger number of data. The improvement in the results of the clustering-based approaches is due to the fact that we can keep the semantics of the data and the actual configuration of each type of feedback at the time of clustering. Thus, unlike the Group-based approach, in which we combine distance matrices, in the proposed approaches we combine the groups of users generated by clustering, through data clustering techniques that allow us to combine groups of users from different information.

Finally, most of the results showed improvement in the accuracy of the recommendation, especially when using more than one type of feedback in the recommender process. This emphasizes that we can better represent a user's behavior when we have a large number of metadata about him. As shown in the experiments, the approach is flexible and extensible to different combinations of feedback types, clustering and recommender algorithms and datasets, although some of these configurations result in marginal improvements over the baselines (in particular collaborative filtering algorithms).

7. FINAL REMARKS

It is very important for a recommender algorithm to have the capability of making recommendations with high quality by analyzing and retrieving user's preferences. Although collaborative filtering is largely used in recommender systems, some efforts to overcome its drawbacks have to be made to improve the prediction accuracy. Selecting the similar users plays an important role on improving the prediction quality. In this article, we extend our previous work, called Group-based Collaborative Filtering [da Costa et al. 2016], to improve the quality of groups generated through two clustering ensemble approaches. The first ensemble clustering technique is based on heuristics, which combines individual results of each feedback generated by clustering algorithm using vote strategy, where the most common label in the result is attributed to the final result. For a more reliable and robust tool, the second technique was used based on CSPA, HGPA and MCLA, called Consensus Clustering Based Strategy, in which user's label is defined by well-known clustering ensembles strategy.

We conducted experiments in two datasets and the results show that our ensemble strategies improve the overall system's performance and make progress in the data clustering in recommender systems. The main advantage of our approach is the possibility of providing more accurate recommendations, consequently reducing the effects of sparsity, since the approach enriches the recommendation process with different types of feedback based on ensemble clustering approaches.

As future work, we plan to evaluate our approach with additional datasets from other domains in order to check the accuracy with different information types. Furthermore, we plan to consider community detection in graphs to select better users' groups to make the recommendation.

REFERENCES

Aggarwal, C. C. *Recommender Systems: The Textbook.* Springer Publishing Company, Incorporated, 2016.

Bobadilla, J., Ortega, F., Hernando, A., and GutiéRrez, A. Recommender systems survey. *Knowledge-Based Systems. http://dx.doi.org/10.1016/j.knosys.2013.03.012* vol. 46, pp. 109–132, July, 2013.

Breese, J. S., Heckerman, D., and Kadie, C. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence. http://dl.acm.org/citation.cfm?id=2074094.2074100.* UAI'98. pp. 43–52, 1998.

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems.* RecSys 2011. ACM, New York, NY, USA, 2011.

da Costa, A. F., Manzato, M. G., and Campello, R. J. Group-based collaborative filtering supported by multiple users' feedback to improve personalized ranking. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web. http://doi.acm.org/10.1145/2976796.2976852.* Webmedia '16. ACM, New York, NY, USA, pp. 279–286, 2016.

da Costa Fortes, A. and Manzato, M. G. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web.* WebMedia '14. ACM, New York, NY, USA, pp. 47–54, 2014.

da Costa Fortes, A. and Manzato, M. G. Case recommender: A recommender framework. In *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web.* Webmedia '16. SBC, pp. 99–102, 2016.

Estivill-Castro, V. Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.* 4 (1): 65–75, June, 2002.

Gantner, Z., Rendle, S., Freudenthaler, C., and Schmidt-Thieme, L. MyMediaLite: A free recommender system library. In *Proceedings of the Fifth ACM Conference on Recommender Systems.* RecSys '11. ACM, New York, NY, USA, pp. 305–308, 2011.

Gantz, J. and Reinsel, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east, 2012.

Gupta, U. and Patil, N. Recommender system based on hierarchical clustering algorithm chameleon. In *2015 IEEE International Advance Computing Conference (IACC).* pp. 1006–1010, 2015.

Han, J., Kamber, M., and Tung, A. K. H. Spatial clustering methods in data mining: A survey. In *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, H. J. Miller and J. Han (Eds.). Taylor and Francis, 2001.

Katarya, R. and Verma, O. P. An effective collaborative movie recommender system with cuckoo search. *Egyptian Informatics Journal*, 2016.

Kim, T.-H., Ryu, Y.-S., Park, S.-I., and Yang, S.-B. An improved recommendation algorithm in collaborative filtering. In *Proceedings of the Third International Conference on E-Commerce and Web Technologies.* EC-WEB '02. Springer-Verlag, London, UK, UK, pp. 254–261, 2002.

Koren, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 4 (1): 1–24, Jan., 2010.

Li, W. and He, W. An improved collaborative filtering approach based on user ranking and item clustering. In *Internet and Distributed Computing Systems*, M. Pathan, G. Wei, and G. Fortino (Eds.). Lecture Notes in Computer Science, vol. 8223. Springer, pp. 134–144, 2013.

Mitchell, T. M. *Machine Learning.* McGraw-Hill, Inc., New York, NY, USA, 1997.

Park, Hae-Sang, J.-S. L. and Jun, C.-H. A k-means-like algorithm for k-medoids clustering and its performance. *Proceedings of ICCIE (2006)*, 2006.

Peska, L. Using the context of user feedback in recommender systems. In *Proceedings 11th Doctoral Workshop on Mathematical and Engineering Methods in Computer Science, MEMICS 2016, Telč, Czech Republic, 21st-23rd October 2016*. pp. 1–12, 2016.

Punera, K. and Ghosh, J. Consensus-based ensembles of soft clusterings. *Applied Artificial Intelligence* 22 (7-8): 780–810, 2008.

Rendle, S. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.* 3 (3): 57:1–57:22, May, 2012.

Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. Bpr: Bayesian personalized ranking from implicit feedback. *CoRR* vol. abs/1205.2618, 2012.

Strehl, A. and Ghosh, J. Cluster ensembles — a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* vol. 3, pp. 583–617, Mar., 2003.

Voorhees, E. M. and Harman, D. K. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

Wen, J. and Zhou, W. An improved item-based collaborative filtering algorithm based on clustering method. In *Journal of Computational Information Systems*, M. Pathan, G. Wei, and G. Fortino (Eds.). Lecture Notes in Computer Science, vol. 8. Springer Berlin Heidelberg, pp. 571–578, 2012.