Information Processing and Management 49 (2013) 688–697

Contents lists available at SciVerse ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Cluster searching strategies for collaborative recommendation systems

Ismail Sengor Altingovde [a,*,1], Özlem Nurcan Subakan [b,1], Özgür Ulusoy [c]

[a] L3S Research Center, Hannover, Germany
[b] Department of Computer & Information Science & Engineering, University of Florida, USA
[c] Computer Engineering Department, Bilkent University, Ankara, Turkey

ARTICLE INFO

Article history:
Available online 3 November 2012

Keywords:
Collaborative filtering
Clustering
Inverted index

ABSTRACT

In-memory nearest neighbor computation is a typical collaborative filtering approach for high recommendation accuracy. However, this approach is not scalable given the huge number of customers and items in typical commercial applications. Cluster-based collaborative filtering techniques can be a remedy for the efficiency problem, but they usually provide relatively lower accuracy figures, since they may become over-generalized and produce less-personalized recommendations. Our research explores an individualistic strategy which initially clusters the users and then exploits the members within clusters, but not just the cluster representatives, during the recommendation generation stage. We provide an efficient implementation of this strategy by adapting a specifically tailored cluster-skipping inverted index structure. Experimental results reveal that the individualistic strategy with the cluster-skipping index is a good compromise that yields high accuracy and reasonable scalability figures.

1. Introduction

Collaborative filtering (CF) is one of the most widely encountered techniques in generating recommendations, which formalizes the notion of "word of mouth" in daily life. Ranging from non-profit Web sites to highly competitive e-commerce giants, players of digital society investigate a lot on this technology, so that they can identify and present the most interesting, relevant or enjoyable items, say a movie, album, book, news story, radio/television show, research paper or even friends (in social networking applications), for their users.

In a nutshell, a CF system is based on determining and aggregating the "votes" of like-minded users for an "active user", so that it can make useful recommendations to the active user (Adomavicius & Tuzhilin, 2005). It works over a database of ratings for the items provided by users, either explicitly (e.g., by voting for a movie) or implicitly (e.g., by clicking on a particular hyperlink). The logic behind collaborative filtering systems is that each user belongs to a community of like-minded people; hence the items favored by these users can be used to form predictions or suggestions. A prediction is the system's opinion for an item that is explicitly asked by the user (e.g., should I go to the movie "Harry Potter and the Deathly Hallows"?) and usually expressed in the form of a score in the same scale with the users' ratings. A suggestion list is a ranked list of items that the system believes the user may be interested in. Collaborative filtering has been successfully used in domains such as recommending movies or songs, where the information content is not easily parse-able and traditional information filtering techniques are difficult to apply (see Adomavicius & Tuzhilin, 2005 for an exhaustive survey).

* Corresponding author.
E-mail address: ismaila@cs.bilkent.edu.tr (I.S. Altingovde).
[1] Work done while the author was at Bilkent University.

0306-4573/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.ipm.2012.07.008

In the literature, CF methods using clustering are well-understood to improve scalability, however the findings in terms of the recommendation accuracy are somewhat inconclusive. In many works (e.g., Breese, Heckerman, & Kadie, 1998; Linden, Smith, & York, 2003; Sarwar, Karypis, Konstan, & Riedl, 2002), once the clusters of users are formed, neighborhood computation stage compares the active user with each of the cluster representatives (i.e., a virtual user including aggregated votes of all users in the cluster) to obtain the most similar clusters. In the next step, the entire clusters (or, equivalently the representatives) are used to generate the recommendation. However, the results reported in some other (e.g., Xue et al., 2005) studies reveal that, once most similar clusters are determined, it is better – in terms of accuracy – to "look into" these clusters to retrieve the most similar users from the clusters. And then, the recommendations can be generated by aggregating the votes of these "real" users but not the cluster representatives. In this paper, we call the former approach *aggregating strategy* and the latter *individualistic strategy*, to denote how clusters are actually used for recommendation generation. Being a more accurate approach, the individualistic strategy requires user-by-user comparisons for each similar cluster, which can hurt the scalability improvements provided by clustering, if done in a straightforward manner. Note that, an alternative to clustering users is constructing item clusters for the item-similarity based CF approach (Sarwar, Karypis, Konstan, & Riedl, 2001). Methods described in our study are potentially applicable to this case, as well.

In this paper, we focus our attention on *cluster-based* CF. As a contribution; we present a specially tailored inverted index structure originally proposed for information retrieval over clustered collections (Altingovde, Can, & Ulusoy, 2006; Altingovde, Demir, Can, & Ulusoy, 2008; Can, Altingovde, & Demir, 2004) and adapt it into the CF framework. The, so-called, *cluster-skipping* inverted index is intended to allow applying the individualistic strategy without too much degradation in the efficiency. That is, the proposed index structure makes it possible to look into the user clusters during the neighborhood formation stage so that the "actual users" can form the neighborhood and subsequently be used during recommendation generation, instead of the virtual ones, i.e., cluster representatives. We envision that this strategy would improve accuracy of cluster-based CF methods, while retaining their advantages in terms of scalability. We further investigate the performance of the proposed approach for cluster-based *hybrid filtering* (HF) systems, which makes use of content-based features, as well. Our experiments based on a large publicly available dataset justify the use of cluster-skipping index for cluster-based CF scenarios.

The rest of the paper is organized as follows: Section 2 presents related work on CF, with special emphasis on the works about cluster-based CF. In Section 3, we describe a baseline CF system, which is based on the *k*-nearest-neighbor approach and employs an inverted index for improving scalability (as proposed in Cöster and Svensson (2002)). We discuss our adaptation of a special inverted index structure for efficient and accurate cluster-based CF in Section 4. Next, in Section 5, we present a cluster-based HF system, which combines content-based filtering with CF by a two-stage clustering, i.e., first clustering the items and then imposing user clusters on top of the item clusters. Experimental results are presented in Section 6. Section 7 concludes the paper and points future work directions.

2. Related work

Collaborative filtering (CF) and content-based recommendation are two important classes of recommender systems. In a recommender system, a set of users are expected to rate a set of items (Su & Khoshgoftaar, 2009; Zhan et al., 2010). CF takes the rating values into account to make recommendation, while content-based recommender systems use the features of users and items (Li & Jin, 2003; Su & Khoshgoftaar, 2009). Hybrid approaches also exist combining the features of CF and content-based recommendation (Burke, 2002; Choi, Jeong, & Jeong, 2010; de Campos, Fernández-Luna, Huete, & Rueda-Morales, 2010; Li, Myaeng, & Kim, 2007). More recently, in addition to the users and items, external ratings (Umyarov & Tuzhilin, 2011) (aggregated from the external sources) and contextual information (such as time and place) that can be obtained either explicitly, implicitly or via data mining techniques are also incorporated into recommendation process (Adomavicius, Mobasher, & Ricci, 2011; Palmisano, Tuzhilin, & Gorgoglione, 2008).

Traditional CF algorithms can be classified into memory-based and model-based algorithms (Breese et al., 1998). For memory-based CF systems, generating a recommendation for an active user involves two stages. During neighborhood formation stage, the system finds the other users (typically at most k of them) that are most similar to the active user with respect to their votes on common items, i.e., similar likes and dislikes. During the actual recommendation stage, the system aggregates the information from the user's neighborhood and either provides a prediction score for some particular item or outputs a recommendation (suggestion) list of promising unseen items for the active user. Subsequently, memory based approaches can be equivalently called *k*-nearest-neighbor (*k*-NN) collaborative filters.

k-NN computation is dynamic and immediately reacts to changes in the user database. Every new rating added to the user database is included in the neighborhood calculation, since similarities between users are calculated in memory when needed. On the other hand, this dynamicity leads to an important drawback for these systems. The necessity of comparing all users to an active user on the fly creates a significant efficiency bottleneck for large systems with hundreds of thousands of users, tens of thousands of items and millions of rating scores. So, scalability is an important handicap of pure *k*-NN approaches. Several optimizations for improving *k*-NN performance were proposed, such as precomputing neighborhoods and sampling (Adomavicius & Tuzhilin, 2005).

As an alternative approach to solve the scalability problem, model-based collaborative filtering algorithms develop an internal model of the available user ratings database by using approaches that are widely employed in statistics and machine

learning, such as Bayesian networks, neural networks, probabilistic relevance, clustering and rule-based methods (e.g., Breese et al., 1998; Greinemr, Su, Shen, & Zhou, 2005; Wang, de Vries, & Reinders, 2008; Wang, Robertson, de Vries, & Reinders, 2008). These systems use a probabilistic approach and compute the expected value of a user prediction, given his/her ratings on other items. In terms of accuracy, model-based algorithms can perform as well as memory-based ones (Breese et al., 1998). Model-based algorithms require more time to train, which is typically off-line, but do not suffer from memory bottlenecks and can provide predictions in shorter time in comparison to memory-based algorithms. However, the inherent static structure of these techniques makes it difficult to update the model without rebuilding it.

While some approaches as exemplified above attempt to improve the efficiency of *k*-NN based CF, an alternative line of research is providing model-based algorithms that are both scalable and as accurate as *k*-NN. To this end, clustering users is a widely employed approach, either directly or as a preprocessing stage (Adomavicius & Tuzhilin, 2005; Gong, 2010; Pham, Cao, Klamma, & Jarke, 2011; Truong, Ishikawa, & Honiden, 2007; Zhang & Hurley, 2009).

In one of the earliest works, Ungar and Foster developed a statistical model for CF and employed clustering for estimating model parameters (Ungar & Foster, 1998). Their findings using repeated clustering (i.e., based on both content-based features and user ratings) are very promising. However, they also note that as clustering smears-out data, it may lead to overgeneralization. Breese et al. also experiment with clustering algorithms and conclude that in many cases clustering methods lead to inferior accuracy with respect to memory-based approaches (Breese et al., 1998). This is usually attributed to the observation that clustering techniques produce less-personal recommendations than other methods. A similar observation is also stated by Sarwar et al. (2002). In that study, authors cluster the users of the CF system by applying the bisecting *k*-Means algorithm. During recommendation generation, they select the most similar cluster as the "neighborhood" for an active user and compute the recommendation as an aggregation of all members of that cluster. Finally, similar arguments appear in an industrial report discussing the item-to-item CF system employed at Amazon.com (Linden et al., 2003). In particular, Linden et al. compare their item-based algorithm with a CF approach utilizing user clusters. They conclude that using clustering for CF improves scalability, but again generating recommendations based on all members of a cluster degrades accuracy. Increasing the number of clusters, on the other hand, diminishes the gains in term of efficiency.

Remarkably, in all of these works, the recommendations are generated by using the entire cluster(s) as the neighborhood or by using just the cluster representative, which is again nothing but an aggregation of the votes of all members in a cluster. Clearly, using the entire cluster (or, equivalently, its representative) as the actual neighborhood significantly improves scalability, as the comparisons are only between cluster representatives and the active user. The number of comparisons can be even smaller in the order of a few magnitudes. But this is also the cause of less-personalized recommendations, as the entire cluster is used to generate final recommendations. Recall that, in this paper, we call this approach the *aggregating strategy*, implying that the clusters are used for neighborhood formation and recommendation generation in their entirety.

In a latter work, Xue et al. exploit user clusters in two ways (Xue et al., 2005). First, they use clusters for smoothing purposes, i.e., to generate some "made-up" ratings for missing ones in the cluster members. Second, they identify most similar clusters to an active user, to obtain a candidate neighborhood, and then individually compare all members of these clusters to the active user to obtain the final neighborhood. In this paper, we call this approach the *individualistic strategy*, implying that recommendations are based on the profiles of individual members of clusters. Experiments show that, if the selected clusters cover around 40% of the whole database, it is possible to cover up to 95% of neighbors that would be obtained by a straightforward *k*-NN algorithm (Xue et al., 2005). In this case, accuracy is almost the same as *k*-NN, whereas savings in efficiency is non-trivial.

Investigating the earlier work reveals that using clustering in CF can yield in several advantages, such as

- improving efficiency with respect to memory-based approaches (Adomavicius & Tuzhilin, 2005; Sarwar et al., 2002; Xue et al., 2005),
- allowing smoothing for handling the missing values and thus remedying data sparsity problem (Xue et al., 2005),
- providing stability and robustness against the profile injection attacks (Mobasher, Burke, & Sandvig, 2006), and
- providing almost the same accuracy figures as *k*-NN, if tuned appropriately (Xue et al., 2005).

The final benefit in the above list, comparable accuracy, is usually obtained when clusters are used to reduce the search space for neighborhood but not to define the neighborhood itself. That is, the individualistic strategy should be applied. However, given that the number of clusters is small with respect to users, which is typically the case, and a certain fraction of data should be covered (e.g., up to 40% as discussed in an earlier study (Xue et al., 2005)) for comparable accuracy, a method is required for efficiently comparing the active user with the cluster members. This is where our solution fits. In particular, we propose to use an inverted index for achieving user–user comparisons within each cluster instead of the entire data space. Of course, it is infeasible to create a separate index for each cluster, and that is why we adapt an inverted index that is specifically tailored for clustered document collections (Altingovde et al., 2006, 2008; Can et al., 2004).

Finally note that, our work belongs to the class of studies (such as Bellogín, Wang, & Castells, 2011; Cöster & Svensson, 2002; Wang, de Vries, et al., 2008) that propose to adapt solutions from information retrieval area, which is inherently related to collaborative filtering. Indeed, our baseline strategy described in the next section is based on the work of Cöster and Svensson (2002), which adapts a typical inverted index structure to CF framework to improve the efficiency of the neighborhood formation process.

3. Baseline collaborative filtering system

In the area of information retrieval (IR), the inverted index is known to be the most common and efficient data structure to retrieve the documents that are most similar to a given user query. Noticeably, the problem of finding documents that are similar to a given query has much in common with the problem we face in collaborative filtering: in both cases, the data space is high dimensional (number of terms and users/items may be tens of millions, or even more) but the individual vectors (representing documents and users) are quite sparse, and both applications require finding most similar vectors to a given vector using some similarity metric. Thus, it seems reasonable to use an inverted index during the neighborhood computation stage of collaborative filtering. Indeed, in a previous study, this approach was employed and found to be promising (Cöster & Svensson, 2002). In that work, the inverted index is stored on the disk and nearest neighbors to an active user are computed by using this index. While computing the similarity between an active user and all other users, several similarity measures, such as Pearson Correlation Coefficient (PCC) and Default Voting, are experimented. These measures can be coupled with inverse user frequency, another well-explored idea from information retrieval literature, which assigns higher weights to items that are more rarely voted, and vice versa (Adomavicius & Tuzhilin, 2005). To compute these similarity functions, there needs to be more than one accumulator structure in the main memory, each of which keeps some part of the partial similarity of each user in the database to the active user. In this paper, we use PCC for computing the similarity between any two users a and i, as shown in the Eq. (1) below (Breese et al., 1998).

$$w(a,i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} \tag{1}$$

where v_i is the profile (i.e., all previous item ratings) of the user i, $v_{i,j}$ is the user i's rating for item j, $\bar{v}_i$ is the mean value of the all ratings of the user i, and j is the running index on the intersecting items rated by the users a and i.

Using an inverted index allows us having as many users and items as possible, without being bounded by memory size, but also enables us to compute exactly the same neighborhood with the typical k-NN approach. Inverted index provides considerable scalability while computing the neighborhood, but in turn it has an update cost, just like the other model-based approaches. Nevertheless, there are several well researched update algorithms for inverted index files, which could be devised in the CF framework, as well. As stated by Cöster and Svensson, the inverted index also allows several optimization opportunities, such as the early pruning strategies like *quit* and *continue* to improve scalability, and index compression techniques, to improve storage efficiency (Cöster & Svensson, 2002).

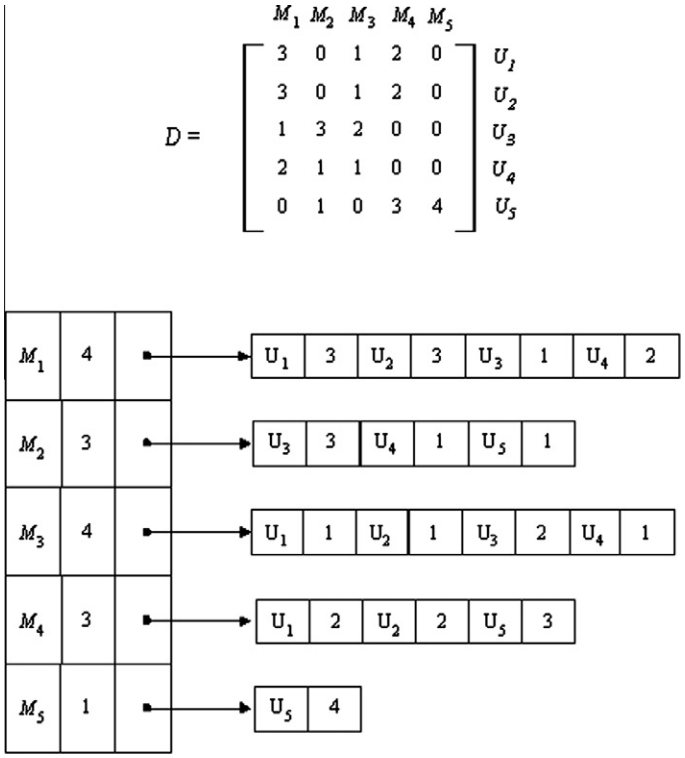

Fig. 1. A user-movie database and corresponding inverted index.

In Fig. 1, we illustrate an example ratings database of users and movies and its corresponding inverted index file, as implemented in our framework. The header of the index, i.e., the list of movies and pointers to posting lists, is kept in the main memory. For each movie, the corresponding posting list captures <user-id, rating> pairs for those users who have seen and rated this movie. Typically, posting lists reside on the disk. During recommendation generation, say for an active user U_a with the profile $\{\langle M_2, 2\rangle, \langle M_4, 5\rangle\}$, the posting lists of the inverted index are read into the memory for each of the items M_2 and M_4. For each such list, partial similarity scores among the active user and users that appear in the posting lists are computed by Pearson Correlation Coefficient shown in Eq. (1). That is, while processing the posting list of M_2, the accumulator entries for users U_3, U_4 and U_5 capture the similarity score of U_a to these users. While processing the list of M_5, accumulator of U_5 is updated once more. After all posting lists corresponding to the items rated by the active user are processed; the accumulators store the actual similarity of the active user to those users who have rated at least one common movie with U_a. These accumulators are sorted and top-k users are extracted, and thus the neighborhood is formed. Finally, the profiles of these users are aggregated to obtain a prediction for a single given item or a suggestion list for the active user.

In this paper, we call the above described k-NN based approach with an inverted index as the *baseline* CF system. This system is used for providing efficiency and accuracy comparisons with respect to the cluster-based filtering systems described in the following.

4. Cluster-skipping inverted index for collaborative filtering

One main contribution in this paper is the adaptation of cluster-skipping inverted index structure for efficient and accurate cluster-based collaborative filtering. This data structure is originally proposed for efficient cluster-based retrieval in the field of information retrieval (Altingovde et al., 2006, 2008; Can et al., 2004).

In Fig. 2, we illustrate the cluster-skipping inverted index for the rating matrix provided in Fig. 1. As in a typical index, each posting list *header* contains the associated movie, the number of posting list elements associated with that movie, and the posting list pointer (disk address). However, the difference from a typical index is in the following. The posting lists include two types of elements, namely <cluster-id, position of the next cluster> pairs in addition to the usual <user-id, rating> pairs. For instance, suppose that users of Fig. 1 are clustered as follows: $C_1 = \{U_1, U_2\}$, $C_2 = \{U_3, U_4\}$, and $C_3 = \{U_5\}$. In this case, posting list of movie M_2 would include three typical posting elements corresponding to the users U_3, U_4 and U_5, as before. In our index structure, the users from the same clusters are grouped together in the posting list, and at the beginning of each such group there is an additional element denoting the cluster of the succeeding users and a pointer to the next group. For instance, the first element in the posting list of M_2 is <C_2, pointer> which means that all the users – until the element pointed by the address – are from cluster C_2, which happens to be users U_3 and U_4. The second additional element <C_3, nil> states that all succeeding users are from cluster C_3 (e.g., only user U_5) and there are no more clusters, i.e., the list ends.

Neighborhood formation for individualistic strategy using this index is achieved as follows. First, for an active user U_a, most similar, say top-N, clusters are determined, typically by comparing cluster representatives to the profile of U_a. We call

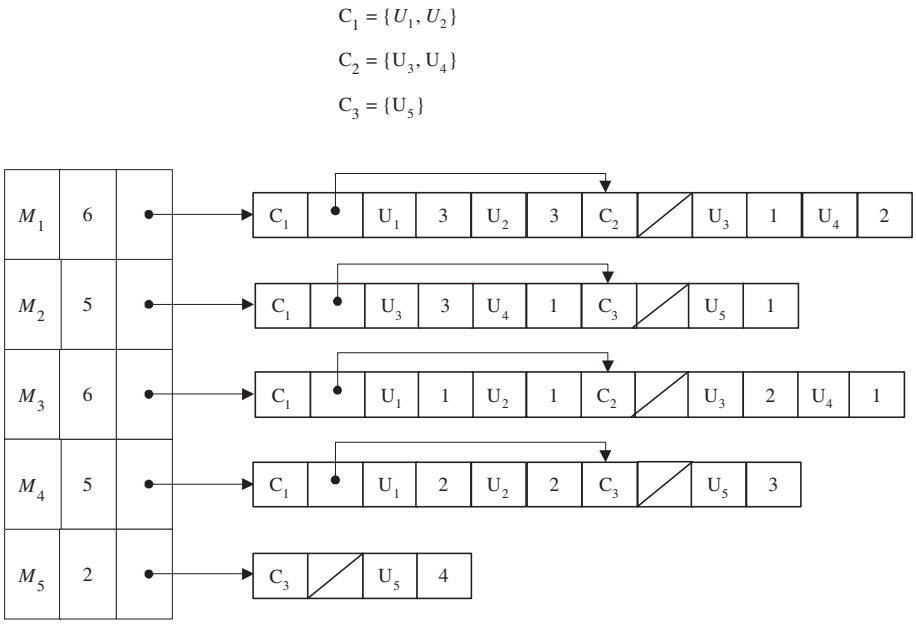

Fig. 2. A cluster-skipping inverted index for the sample user-movie database.

these clusters *best-clusters*. Next, for each item rated by U_a, the corresponding posting list is retrieved. However, we update the partial similarities stored in the accumulators for only those users that are from the best-clusters. Thus, only those users that are within the best-clusters are considered for neighborhood formation. When all the items rated by the active user are processed, the users are sorted with respect to similarity scores and top-k of them are selected. Once the neighborhood is formed, the recommendation generation stage is the same as in the usual k-NN approach. The pseudo-code for this procedure is given in Fig. 3. Note that, for the aggregating strategy, neighborhood actually corresponds to the best-clusters, so there is no need to use the inverted index, at all.

For instance, assume an active user U_a with the profile $\{\langle M_2,2\rangle,\langle M_4,5\rangle\}$. Also assume that we set the number of best-clusters and the size of user neighborhood (k) to 1, i.e., we are to select the most similar cluster considering the representatives, and then to select the most similar user from this single best-cluster. Assume also that, the best cluster is found to be C_3. While processing the posting list of M_2 we skip the portion corresponding to C_1 (since it is not a best-cluster) and directly jump to the C_3 portion by following the address in the first element. Similarly, while processing the posting list of M_4; we again skip the unnecessary C_1 portion of the posting list and only consider the part corresponding to C_3. In other words, by using the skip approach we only handle the users that are in the best clusters and really need to be compared to the active user. At the end of the process, only the user U_5 has a non-zero similarity score and will be considered for generating recommendation. Note that, in practice, there are more than one accumulator structures storing different components of the PCC metric (Cöster & Svensson, 2002), but we prefer to mention only one accumulator (both in the discussion and in Fig. 3) for the sake of simplicity.

Remarkably, the top-k users obtained at the end of the above procedure are different than those obtained by the baseline system, since only the subset of users that are from the best-clusters are considered. Subsequently, the use of cluster-skipping index is expected to improve the scalability for cluster-based CF, exactly in the same way a typical inverted index improves scalability for k-NN based CF, i.e., our baseline in this paper.

One can argue that after the best-clusters are found, users from the best-clusters can be determined in a succeeding stage by using a typical inverted index (i.e., by first identifying all users that have non-zero similarity to an active user and then eliminating those that are not from the best-clusters). Although this is possible (Altingovde et al., 2006), using a specific cluster-skipping IIS improves performance significantly (Can et al., 2004). Furthermore, the gain is expected to be much higher than the information retrieval case, since typical query size in IR is only a few words whereas the profile of an active user can include hundreds of items. Earlier works (e.g., Can et al., 2004) in IR field reveal that gain of skipping unused portions is highest for the queries with more than a hundred terms, which can be a typical profile size in CF framework. Furthermore, skipping unnecessary segments of posting lists provides crucial savings when posting lists are stored in a compressed format and should be decompressed during similarity comparisons (Altingovde et al., 2008).

5. Cluster-based hybrid filtering (HF)

Hybrid recommender systems combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual recommender technique. Most commonly, collaborative filtering is combined with content based filtering in an attempt to remedy the associated problems of each approach.

Ungar and Foster used a cluster-based hybrid filtering approach in which they first cluster music compact discs (CD) with respect to a content-based feature, e.g., the artist (Ungar & Foster, 1998). In the second pass, they cluster users on those CD clusters. The final clusters are used to create a formal model for collaborative filtering and generate recommendations. We

Input: Active user profile U_a = {<I_1, R_1>, <I_2, R_2>…} where I_i is item i and R_i is the rating for item i, best-clusters B= {C_1, C_2, …C_N}, where C_i is in the top-N most similar clusters to U_a, *header* part of the cluster-skipping inverted index created on the user-item database

In-memory data structures: User-user similarity accumulator structure *Acc*
Disk-based data structures: Posting lists L of the cluster-skipping inverted index created on the user-item database

For each item I_i in U_a
  Retrieve the posting list L_i corresponding to the item I_i
  Access the first list element <*cluster-id, pointer*> from L_i
  If *cluster-id* is in B
    Until the address given by *pointer* is reached
      Access the typical element <user-id U_i, rating UR_i>
      Compute partial similarity *sim* by using UR_i, Ri and a similarity metric (e.g., PCC)
      $Acc\,[U_i] \leftarrow Acc\,[U_i] + sim$
  Else
    Jump to the next cluster given by the *pointer*

Return top-k users that have the highest *Acc* scores

Fig. 3. Neighborhood formation procedure for individualistic strategy using cluster-skipping inverted index.

prefer to call this a hybrid filtering system, since it uses both content-based features and user rating data. The authors report that the two-stage clustering as described above works very well.

As another contribution of this paper, we compare the cluster-based CF with cluster-based HF. We believe that it is worthwhile to evaluate the performance of multi-stage clustering in our framework, which allows efficient computation of individualistic strategy, as well as the generally used aggregating strategy.

Our multi-stage clustering approach first generates the clusters based on content information of items. For our prototype system recommending movies, genre attribute is selected for this purpose. Thus, we obtain clusters such as say, *Action*, *Drama* and *Romance*. These clusters are overlapping, i.e., a movie may fall into several clusters, such as both *Drama* and *Romance*. Then the users are also clustered according to genre of the movies they watched. These clusters are non-overlapping, since a user falls into the cluster from which (s)he has watched the largest number of movies. Then, the users of each cluster obtained in this stage are clustered for the last time, but according to their movie ratings, as before. For instance, the users who have watched movies from *Action* cluster are clustered among themselves, and similarly the users who have watched movies from *Drama* cluster are clustered among themselves. Once the final cluster structure is obtained, both aggregating and individualistic strategies are applicable, as described before.

6. Evaluation

6.1. Data set

We use a large publicly available dataset, EachMovie, as the test-bed for our approaches, similar to some previous studies (e.g., Cöster & Svensson, 2002; Xue et al., 2005). This dataset comprises a total of 2,811,983 ratings of 1628 movies by 60,087 users. Each rating is on a scale of 0–5. We also employ a Web crawler, WebSPHINX, (http://www.cs.cmu.edu/~rcm/websphinx/) developed by Rob Miller at Carnegie Mellon University, in order to gather the content data of movies from The Internet Movie Database (IMDb) (http://www.imdb.com/) for hybrid filtering approaches. Out of 1628 movies present in EachMovie dataset, the content information – including synopsis, genre, key actors and actresses – of 1482 movies could be downloaded. Out of these 1482 movies, 1224 movies have synopsis.

6.2. General parameters

We experiment with a training set size of 90% of all users, i.e., 55,000 users, following the practice in an earlier work for comparison purposes (Cöster & Svensson, 2002). Neighborhood size k is set to 30. We set the number of best-clusters to 10% of the total number of clusters. We employ *AllBut1* protocol, meaning that for each user we held out a single vote that should be predicted on the basis of all the other votes in the profile. For each test user, the evaluated CF system is required to return a prediction score for five movies in that user's profile. Experiments are repeated five times with randomly obtained train/test splits.

6.3. Evaluation metrics

To evaluate accuracy and efficiency of competing approaches, we use *mean absolute error* (MAE) and the elapsed time for neighborhood formations, respectively. MAE is a popular statistical accuracy metric and calculates the absolute difference between the actual vote and the prediction, averaged over all predictions. For our data set, MAE is on a scale of 0–5. Obviously, smaller MAE values imply better accuracy. We measure the total elapsed time for neighborhood formations (NF) for five predictions of all test users. The mean neighborhood formation time (referred to as Mean NF Time) is computed for generating a single prediction for a particular user.

6.4. Clustering parameters

In this paper, we employ two different clustering algorithms – namely, k-Means and C^3M, to obtain user clusters for the cluster-based CF systems and hybrid filtering systems. The well-known k-Means algorithm requires the number of clusters to be provided as an input (e.g., see Manning, Raghavan, & Schütze, 2008). In contrast, C^3M, as one of its most important features, can determine the number of clusters automatically (Can & Ozkarahan, 1990). For both algorithms, off-line clustering of the training dataset is achieved in the order of minutes using commodity hardware and general (un-optimized) implementations of the algorithms. Note that, while we prefer to employ C^3M and k-Means clustering algorithms to be comparable with some previous works (Altingovde et al., 2008; Can et al., 2004; Xue et al., 2005), our strategy can be coupled with any other flat clustering algorithm with a reasonable output quality and algorithmic complexity.

We experiment with three different values for the number of produced clusters – namely, 24, 200, and 1300, for both C^3M and k-Means algorithms. The first value, 24 is the number of clusters that is determined by C^3M automatically. We repeat our experiments for k-Means with the same value, for comparison purposes. Recall that, in the hybrid filtering system, clustering is multi-stage. The users are first clustered on movie genres, a process that does not require a clustering algorithm, and then re-clustered according to their ratings' similarities. 200 is the number of clusters obtained by C^3M for this second clustering

stage. For the final setting, we manually fix the number of second-stage user clusters to be 100 per genre-cluster, and since there are 13 of them (i.e., movies in our data set fall into 13 different genres, as we extracted from IMDB site), the total number of clusters is 1300.

6.5. Evaluated systems

We essentially compare the following three systems as described in this paper.

- Baseline CF system that is based on *k*-NN and employing a typical inverted index.
- Cluster-based CF system.
- Cluster-based HF system.

For the latter two cluster-based systems, we compare performances with aggregating strategy, which generates recommendations from all members of best-clusters, and individualistic strategy, which generates recommendations by inspecting the members of best-clusters individually and determining *k*-nearest neighbors among them. While computing the individualistic strategy, both systems employ a cluster-skipping inverted index. Aggregating strategy, obviously, does not need such an index.

6.6. Experimental results and discussions

In Tables 1 and 2, we provide MAE and neighborhood formation times for the baseline CF system and the cluster-based CF system, using individualistic and aggregating strategies, respectively. Verifying our initial hypothesis, the cluster-based CF system using individualistic strategy performs almost as accurate as the baseline system, regardless of the clustering algorithm and the number of clusters. For instance, baseline CF system achieves an MAE score of 0.91 (repeated in all tables for the ease of comparison). Cluster-based CF system with individualist strategy also achieves MAE scores between 0.90 and 0.92, for varying clustering algorithms and parameters. On the other hand, cluster-based CF with aggregating strategy reaches to an MAE score of 1.18 at the best case and can worsen up to 1.32. Thus, looking inside the best-clusters pays off in terms of accuracy.

In terms of neighborhood formation (NF) time, cluster-based CF with aggregating strategy, which by definition does not need the inverted index, can reduce the required time to only a few percent of the baseline system. However, this gain in performance is obtained in return for a significant sacrifice in the prediction accuracy. As shown in Table 2, the MAE scores for CF with aggregating strategy reflect an increment ranging from 32% up to 47% in comparison to the baseline MAE. On the other hand, cluster-based CF with individualistic strategy can achieve the same accuracy figures with the baseline system, and the overall time for NF is still less than the half of the time required by the baseline. Thus, the individualistic strategy accompanied with cluster-skipping index is a compromise, which provides good accuracy and more than 50% savings in NF Time.

In Tables 3 and 4, we provide MAE and neighborhood formation times for the baseline CF system and the cluster-based HF system, using individualistic and aggregating strategies, respectively. The results clearly follow the same trends discussed in the above. Interestingly, the use of content-based features and repeated clustering does not seem to improve accuracy figures over the simple user-clustering approaches discussed above.

We can draw the following conclusions from the experimental results:

- Our results reveal that, computing the neighborhoods by inspecting the actual users that fall into the best-clusters (i.e., the individualistic strategy) yields higher prediction accuracy in comparison to using the aggregations of these best-clusters (i.e., their representatives) for neighborhood formation. This result also justifies the need for using cluster-skipping inverted index for the cluster-based collaborative filtering task, as an efficient way of implementing above strategy.
- In terms of neighborhood computation time, collaborative filtering with cluster-skipping inverted index takes far more shorter time (in some cases, almost 40%) than collaborative filtering with a typical inverted index, since the number of comparisons is significantly reduced by the use of cluster-skipping index structure.

Table 1
Comparison of baseline CF system vs. cluster-based CF with cluster-skipping inverted index and using individualistic strategy.

System	Baseline CF	Cluster-based CF with cluster-skipping inverted index					
		C^3M			*k*-Means		
No. of clusters		24	200	1300	24	200	1300
Accuracy	0.91	0.92	0.91	0.90	0.92	0.91	0.90
NF Time (s)	1324.57	651.01	593.09	815.15	531.22	883.02	1027.05
Mean NF Time (ms)	49.86	24.15	22.00	30.24	19.71	32.76	41.52

Table 2
Comparison of baseline CF system vs. cluster-based CF using aggregating strategy.

System	Baseline CF	Cluster-based CF					
		C^3M			*k*-Means		
No. of clusters		24	200	1300	24	200	1300
Accuracy	0.91	1.18	1.25	1.29	1.22	1.30	1.32
NF Time (s)	1324.57	81.27	72.06	111.74	70.68	73.79	84.78
Mean NF Time (ms)	49.86	3.02	2.67	4.15	2.62	2.74	2.92

Table 3
Comparison of baseline CF system vs. cluster-based HF with cluster-skipping inverted index and using individualistic strategy.

System	Baseline CF	Cluster-based HF with cluster-skipping inverted index			
		C^3M		*k*-Means	
No. of clusters		184	1300	184	1300
Accuracy	0.91	0.90	0.90	0.94	0.94
NF Time (s)	1324.57	623.82	902.78	829.22	1747.47
Mean NF Time (ms)	49.86	23.33	33.53	30.80	64.90

Table 4
Comparison of baseline CF system vs. cluster-based HF using aggregating strategy.

System	Baseline CF	Cluster-based HF			
		C^3M		*k*-Means	
No. of clusters		184	1300	184	1300
Accuracy	0.91	1.18	1.29	1.32	1.31
NF Time (s)	1324.57	73.13	74.58	73.81	84.52
Mean NF Time (ms)	49.86	2.74	2.77	2.74	2.91

- Interestingly, the cluster-based hybrid filtering approach does not improve the prediction accuracy as it might be expected (i.e., a contradictory result to what is reported in, for instance, (Ungar & Foster, 1998). We attribute this to the fact that in our experiments, the only content attribute used is the "genre" for the initial clustering stage. We anticipate that employing other attributes such as plot and artists can further improve accuracy in this scenario. This remains as a future work issue.

7. Conclusion

Collaborative filtering is one of most important techniques used in the recommendation systems, and continues to gain significant attention from both academic and commercial parties. In this paper, we show that cluster-based collaborative filtering techniques, an important class of CF algorithms, can be made both accurate and scalable, using an individualistic search strategy within clusters and a specifically tailored cluster-skipping inverted index. The proposed strategy substantially reduces the neighborhood formation time (i.e., up to 60%) in comparison to using a typical inverted index file without any adverse effects on the recommendation quality. This means that our solution makes the promises of clustering (such as improving scalability (Adomavicius & Tuzhilin, 2005; Sarwar et al., 2002; Xue et al., 2005) and remedying data sparsity (Xue et al., 2005)) attainable for real-life and large-scale recommendation systems without requiring to sacrifice from the recommendation accuracy.

As a future work, we plan to investigate the performance of discussed strategies under highly dynamic conditions and for item based CF strategies. We also aim to incorporate further clues such as the contextual (Adomavicius et al., 2011; Palmisano et al., 2008) and external information (Umyarov & Tuzhilin, 2011) into our cluster-based collaborative filtering framework.

Acknowledgements

This work is supported by The Scientific and Technical Research Council of Turkey (TÜBİTAK) under the Grant No. 105E024. We thank Tugrul Ince, Simal Ince and H. Hakan Ari for their contributions during the implementation.

References

Adomavicius, G., Mobasher, B., & Ricci, F. (2011). And Alexander Tuzhilin, context-aware recommender systems. *AI Magazine, 32*(3), 67–80.

Adomavicius, G., & Tuzhilin, A. (2005). Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, 17*(6), 734–749.

Altingovde, I. S., Demir, E., Can, F., & Ulusoy, Ö. (2008). Incremental cluster-based retrieval using compressed cluster-skipping inverted files. *ACM Transactions on Information Systems, 26*(3).

Altingovde, I. S., Can, F., & Ulusoy, Ö. (2006). Algorithms for within-cluster searches using inverted files. In *Proceedings of the international symposium on computer and information sciences* (pp. 707–716). Istanbul, Turkey.

Bellogín, A., Wang, J., & Castells, P. (2011). Text retrieval methods for item ranking in collaborative filtering. In *Proceedings of the European conference on IR research* (pp. 301–306). Dublin, Ireland.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the conference on uncertainty in artificial intelligence* (pp. 43–52). Madison, Wisconsin, USA.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction, 12*(4), 331–370.

Can, F., Altingovde, I. S., & Demir, E. (2004). Efficiency and effectiveness of query processing in cluster-based retrieval. *Information Systems, 29*(8), 697–717.

Can, F., & Ozkarahan, E. A. (1990). Concepts and effectiveness of the cover-coefficient-based clustering methodology for text databases. *ACM Transactions on Database Systems, 15*(4), 483–517.

Choi, S. H., Jeong, Y. S., & Jeong, M. K. (2010). A hybrid recommendation method with reduced data for large-scale application. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 40*(5), 557–566.

Cöster, R., & Svensson, M. (2002). Inverted file search algorithms for collaborative filtering. In *Proceedings of the annual international ACM SIGIR conference* (pp. 246–252). Tampere, Finland.

de Campos, L. M., Fernández-Luna, J. M., Huete, J. F., & Rueda-Morales, M. A. (2010). Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks. *International Journal of Approximate Reasoning, 51*(7), 785–799.

Gong, S. (2010). A collaborative filtering recommendation algorithm based on user clustering and item clustering. *Journal of Software, 5*(7), 745–752.

Greinemr, R., Su, X., Shen, B., & Zhou, W. (2005). Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning, 59*(3), 297–322.

Li, S., & Jin, R. (2003). Flexible mixture model for collaborative filtering. In *Proceedings of the international conference on machine learning* (pp. 704–711). Washington, DC, USA.

Li, Q., Myaeng, S. H., & Kim, B. M. (2007). A probabilistic music recommender considering user opinions and audio features. *Information Processing & Management, 43*(2), 473–487.

Linden, G., Smith, B., & York, J. (2003). Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing, 7*(1), 76–80.

Manning, C. D, Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mobasher, B., Burke, R. D., & Sandvig, J. J. (2006). Model-based collaborative filtering as a defense against profile injection attacks. In *Proceedings of the national conference on artificial intelligence*. Boston, Massachusetts, USA.

Palmisano, C., Tuzhilin, A., & Gorgoglione, M. (2008). Using context to improve predictive modeling of customers in personalization applications. *IEEE Transactions on Knowledge and Data Engineering, 20*(11), 1535–1549.

Pham, M. C., Cao, Y., Klamma, R., & Jarke, M. (2011). A clustering approach for collaborative filtering recommendation using social network analysis. *Journal of Universal Computer Science, 17*(4), 583–604.

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the international WWW conference* (pp. 285–295).

Sarwar, B. M., Karypis, G., Konstan, J., & Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the international conference on computer and information technology*. Dhaka, Bangladesh.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009* (article id 421425).

Truong, K., Ishikawa, F., & Honiden, S. (2007). Improving accuracy of recommender system by item clustering. *IEICE – Transactions on Information Systems, E90-D*(9), 1363–1373.

Umyarov, A., & Tuzhilin, A. (2011). Using external aggregate ratings for improving individual recommendations. *ACM Transactions on the Web, 5*(1), 3.

Ungar, L., & Foster, D. (1998). Clustering methods for collaborative filtering. In *Proceedings of the workshop on recommendation systems*. Menlo Park California, USA.

Wang, J., Robertson, S., de Vries, A. P., & Reinders, M. J. T. (2008). Probabilistic relevance ranking for collaborative filtering. *Information Retrieval, 11*(6), 477–497.

Wang, J., de Vries, A. P., & Reinders, M. J. T. (2008). Unified relevance models for rating prediction in collaborative filtering. *ACM Transactions on Information Systems, 26*(3), 1–42.

Xue, G., Lin, C., Yang, Q., Xi, W., Zeng, H., Yu, Y., et al. (2005). Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the annual international ACM SIGIR conference* (pp. 114–121). Salvador, Brazil.

Zhan, J., Hsieh, C. L., Wang, I. C., Hsu, T. S., Lian, C. J., & Wang, D. W. (2010). Privacy-preserving collaborative recommender systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews, 40*(4), 472–476.

Zhang, M., & Hurley, N. (2009). Novel item recommendation by user profile partitioning. In *Proceedings of the IEEE/WIC/ACM international joint conference on web intelligence and intelligent agent technology* (pp. 508–515). Washington, DC, USA.