Improved Recommendations via (More) Collaboration *

Rubi Boim Haim Kaplan Tova Milo Ronitt Rubinfeld
School of Computer Science
Tel-Aviv University
{boim,haimk,milo,ronitt}@post.tau.ac.il

ABSTRACT

We consider in this paper a popular class of recommender systems that are based on Collaborative Filtering (CF for short). CF is the process of predicting customer ratings to items based on previous ratings of (similar) users to (similar) items, and is typically used by a *single organization*, using its *own* customer ratings.

We argue here that a *multi-organization* collaboration, even for organizations operating in different subject domains, can greatly improve the quality of the recommendations that the individual organizations provide to their users. To substantiate this claim, we present C2F (Collaborative CF), a recommender system that retains the simplicity and efficiency of classical CF, while allowing distinct organizations to collaborate and boost their recommendations. C2F employs CF in a distributed fashion that improves the quality of the generated recommendations, while minimizing the amount of data exchanged between the collaborating parties. Key ingredient of the solution are succinct signatures that can be computed locally for items (users) in a given organization and suffice for identifying similar items (users) in the collaborating organizations. We show that the use of such compact signatures not only reduces data exchange but also allows to speed up, by over 50%, the recommendations computation time.

1. INTRODUCTION

Recommender systems are programs that attempt to present to the user a small subset of items (out of a much larger items set), which she is likely to find interesting. With the growing popularity of e-commerce, and the huge variety of items offered by on-line stores, assisting users in identifying relevant items is crucial. Indeed, much research has been devoted recently to developing effective recommendation algorithms, as even very small improvements directly translate into higher income.

In this paper we focus on a popular class of such recommender systems, where recommendations are computed via Collaborative Filtering (CF for short) [3]. CF is the process of predicting users rating to items based on previous ratings of (similar) users to (similar) items. Unlike semantic-based methods [3], CF does not use semantic properties of the items (e.g, manufacturer, color, etc). Instead, it is based on the assumption that users who agreed in the past on item ratings are likely to agree again in the future. Common CF algorithms thus consist of two main steps: (1) choosing the right neighborhood of users (resp. items), and (2) calculating the prediction using some aggregation function on actual ratings provided by the chosen neighborhood.

Previous works on such recommender systems focused mostly on CF performed by a *single* organization over its *own* customer ratings. We argue here that a *multi-organization* collaboration, even for organizations operating in different subject domains, can greatly improve the quality of the recommendations that the individual organizations provide to their users. Assume for instance that `Netflix` - an online movie rental service, `Blockbuster` - a chain of movie rental stores, and `Toys"R"Us` - a toy store chain, wish to collaborate in order to improve the service to their customers. Naturally, each organization has its own database of user ratings and can use it to generate recommendations. But the quality of recommendations may be greatly improved by taking into account information available in the other collaborating organizations. This seems obvious for `Netflix` and `Blockbuster` where the domain items are similar: Correlations between users interest in one movie store may naturally be used to refine recommendations in the other. But correlations between inter-domain items may also exist and can be leveraged: We may discover, e.g., that a large portion of the users who viewed (and liked) "Star Wars" on `Netflix` also bought (and liked) a space-ship model at `Toys"R"Us`, and thus recommend this toy to similar viewers that have not purchased it yet.

To support such collaboration, we present in this paper C2F (Collaborative, Collaborative Filtering), a recommender system that retains the simplicity and efficiency of classical CF, while allowing distinct organizations to collaborate and boost their recommendations. Note that a naive solution that accumulates all data sets into one centralized location (then applies classical CF) is typically not feasible due to the excessive amounts of (constantly updated) data and the independence of the organizations. Instead, C2F employs CF in a distributed fashion that maximizes the quality of the generated recommendations, while minimizing the amount of data exchanged between the collaborating parties. Key ingredient of the solution are succinct signatures that can be computed locally for items (users) in a given organization and suffice for identifying similar items (users) in the collaborating organizations.

We describe here the operation of C2F and present an experiential analysis, performed on real-life data, demonstrating the improvement in recommendations quality that it achieves, relative to those computed when only the local organization data is consid-

*The research has been partially supported by the European Project Mancoosi, the Israel Science Foundation and the US-Israel Binational Science Foundation.

WebDB '10 Indianapolis, IN USA

ered. We also show that the use of compact signatures not only reduces data exchange but also allows to speed up, by over 50%, the recommendations computation time.

We note that the aggregating of rating predictions from multiple sites has been previously proposed in the literature in a different context: For *semantically-similar* sites, where a given [user,item] pair belongs to multiple sites, it was suggested to compute its rate prediction by aggregating the predictions of the individual sites for this pair [7]. A main difference with our work is their focus on similar domains with common (user,item) pairs. In contrast, our solution allows for *inter-domain* collaboration, exchanging only a small amount of data between the collaborating parties, yet sufficient for effectively identifying and exploiting the relevant information.

It should be stressed that our aim here is *not to invent yet another CF algorithm*, but rather to present a generic novel technique that allows one to better exploit existing CF algorithms in a distributed, multi-organization setting. Indeed, while our implementation uses specific neighborhood metric/rate aggregation functions to generate recommendations, the technique that we propose can similarly be used for other functions.

The paper is organized as follows. Section 2 provides the necessary background on classical CF and presents the adjustments required to support a distributed setting. Section 3 presents our algorithms and Section 4 describes the system implementation and experiments. Finally, Section 5 concludes with related work.

2. PRELIMINARIES

We start with some background on classical CF, then explain the adjustments required to support a distributed setting and present our algorithms.

2.1 Classical CF

Given a set U of users and a set I of items, we use u, v to denote users in U and i, j to denote items in I. A rating $r_{u,i}$ is a numeric value given by a user u to an item i. The ratings are given in the form of a matrix R of size $|U| \times |I|$. Note that this matrix is typically very sparse (usually less than 1% of the possible ratings are known) and the goal is to predict the unknown $r_{u,i}$ ratings as accurately as possible. We consider two popular types of CF prediction algorithms, referred to as *Item-Based* and *User-Based* CF.

Item-Based CF is based on the intuition that an unknown rating $r_{u,i}$ can be estimated using the actual ratings that u gave to items j similar to i. Formally,

$$r_{u,i} = \frac{\sum_{j \in N(i;u)} s_{i,j} r_{u,j}}{\sum_{j \in N(i;u)} s_{i,j}} \quad (1)$$

where $N(i;u)$ is the set (neighborhood) of items similar to i that were rated by u (i.e. $r_{u,j}$ is known for each $j \in N(i;u)$), and $s_{i,j}$ is a measure of the similarity between items i and j. Note that the similarity between different items is used not only to weight the ratings, but also to choose the neighborhood $N(i;u)$. For example, we could choose $N(i;u)$ as the set of items whose similarity value to i is above some given threshold.

User-Based CF is analogous to item-based, but instead of looking at similar items we look at users v similar to u, which have rated i. The prediction of $r_{u,i}$ is then symmetrically obtained by aggregating their ratings for i.

In both approaches, similarity between two items (users) plays a key role in determining the neighborhoods and for weighting the existing ratings. In principle, any similarity measure can be used here (e.g. Cosine or L^i distance) but Pearson's Correlation Coefficient [13] seems to be the preferred choice in most major systems. The basic intuition behind Pearson's measure is to give a high similarity score for two items (users) that tend to be rated the same by many users (resp., tend to rate the same many items). Formally, for two items i and j, it is computed as follows.

$$s_{i,j} = \frac{\sum_{u \in U_{(i,j)}} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U_{(i,j)}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{u \in U_{(i,j)}} (r_{u,j} - \bar{r}_u)^2}} \quad (2)$$

where $U_{(i,j)}$ is the set of all users u who rated both items i and j (i.e. $r_{u,i}$ and $r_{u,j}$ are both known), and $\bar{r}_u$ is the average rating of user u. $\bar{r}_u$ is subtracted here from $r_{u,i}$ to "normalize" the rating values for those users that tend to always give high (or low) grades. When $U_{(i,j)} = \emptyset$, namely there are no users that rated both i and j, we define $s_{i,j} = 0$.

Pearson similarity of two users u, v is computed symmetrically.

2.2 Collaborative CF

Consider an organization o that wishes to improve its rating predictions using information available in another organization o'. For o and o' to collaborate, they should have some common denominator w.r.t. which they can share information. We focus on two scenarios here: The first, called *shared users*, assumes that o and o' share a (relatively large) set of common users. The examples we have seen earlier of `Netflix`, `Blockbuster` and `Toys"R"Us` comply to this scenario, since many customers shop in several of these stores. The second scenario, called *shared items*, is the case where o and o' share a (relatively large) set of common items. For example, assume that `Blockbuster` branches in different countries maintain individual local information repositories. Two branches o, o' here serve different customers (users) but may carry many common movies (items), e.g. American films.

We next explain how shared users (resp. items) can be used to boost Item-based (User-based) CF. We start with some intuition, then explain the optimizations required to make things practical.

We denote below by O the set of collaborating organizations and use o, o' to denote individual organizations in O. A rating of user u to item i in organization o is denoted $r^o_{u,i}$. Finally, the similarity between two items i in o ,j in o' (where o may be the same or different than o') is denoted $s^{(o,o')}_{i,j}$; similarly, $s^{(o,o')}_{u,v}$ denotes the similarity between two users u in o, v in o'.

Item-based CF with shared users is a generalization of the centralized version: Instead of considering only the local neighborhood of items, we improve the prediction by considering also the corresponding neighborhoods in the collaborating organizations. Namely, equation (1) is refined as follows

$$r^o_{u,i} = \frac{\sum_{o' \in O} \sum_{j \in N^{o'}(i;u)} s^{(o,o')}_{i,j} r^{o'}_{u,j}}{\sum_{o' \in O} \sum_{j \in N^{o'}(i;u)} s^{(o,o')}_{i,j}} \quad (3)$$

where $N^{o'}(i;u)$ is the set (neighborhood) of items similar to i *in organization* o' that have been rated by u. As before, the similarity between items is used not only to weight the final prediction, but also to choose the neighborhood.

The similarity to an item j that resides locally in o can be computed as before. Things are more complex for items j that reside in some remote organizations o'. Observe that every item can essentially be viewed as a vector of ratings, in a multi-dimensional space, where each dimension corresponds to a specific user, and holds her rating. Similarity between two items i and j thus amounts to computing the similarity of the two corresponding vectors (w.r.t. the similarity measure of interest.) A straightforward solution is thus to send i's vector to every site $o' \in O$, and have each such site compute and return its $\sum_{j \in N^{o'}(i;u)} s^{(o,o')}_{i,j} r^{o'}_{u,j}$ and $\sum_{j \in N^{o'}(i;u)} s^{(o,o')}_{i,j}$ values (i's vector is used to compute the similarly of items in o' to

i, thereby determining both the neighborhood set $N^{o'}(i;u)$ and the similarity values $s_{i,j}^{(o,o')}$ for the neighborhood members).

It is important to note however is that these vectors are very large (their size equals the number or users that rated a given item, which can be hundreds of thousands [6]). To minimize communication, C2F sends, instead, a concise, much smaller vector (which we call *a signature*) that suffices for identifying similar items in the collaborating organizations. We will see that a further positive result of use of concise signatures is a speed up of over 50% in the recommendations computation time.

The construction of signatures is explained in the next section. But before we explain this, let us first briefly consider the symmetric User-based scenario.

User-based CF with shared items works analogously. Each user is viewed as a vector of ratings, where the dimensions correspond to items and hold the user's ratings for the items. Here again, rather than sending the full vector of the user, C2F sends a smaller signature that suffices for identifying similar users in the collaborating organizations.

3. COMPUTING SIGNATURES

C2F employs two main algorithms for computing users (items) signature, inspired, resp., by works on Dimension Reduction [4] and Features Selection [11]. We explain them below.

3.1 Dimension Reduction

Dimension reduction is a common technique where a set of points (represented by vectors) in a given multi-dimensional space is mapped to a lower-dimensional space, by multiplication with a special matrix M [10]. M is chosen so that the distance (similarity) between the input vectors (measured, e.g., by Cosine or L^i) is preserved up to a small constant.

Recall that, in our setting, every item (resp. user) can be viewed as a vector of ratings, in a multi-dimensional space, where each dimension corresponds to a specific user (item), and holds her (its) rating. Specifically, consider a set O of collaborating organizations. Let us consider first a shared-users scenario (the shared-items case is handled symmetrically, see below). For simplicity, assume that they have exactly the same set U of users. [1] Each organization $o \in O$ handles a set I_o of items and its ranking matrix R_o is thus of arity $|I_o| \times |U|$. In other words, we have $|I_o|$ item vectors, in a $|U|$ dimensional space. If all organizations use the same matrix M of arity $k \times |U|$ (for some small $k < |U|$) to reduce the dimensions of their data, the reduced k-ary vectors could then be exchanged between the sites, instead of the original ones, and used to identify similar items/users.

More formally, consider the reduced matrices $\hat{R}_o = R_o \times M$, $o \in O$ (of arity $|I_o| \times |k|$). Suppose that site o wants to estimate what rank user u would give to item i (denote above $r_{u,i}^o$). Rather than sending the original U-ary vector $R_o[i]$ to the collaborating organizations, o sends the reduced k-ary vector $\hat{R}_o[i]$ ($= R_o[i] \times M$). Now, a collaborating organization o' may compute the similarity value $s_{i,j}^{(o,o')}$ between i and the items j in o' by measuring the the distance between $\hat{R}_o[i]$ and $\hat{R}'_o[j]$. The remaining of the computations continues as before.

Similarly, in a shared-items scenario, each organization $o \in O$ handles a set U_o of users and the ranking matrix R_o is of arity $|U_o| \times |I|$ (rows now corresponds to users). M here is of arity $k \times |I|$. The rest stays as above.

[1] Otherwise, let U denote the union of the users sets.

Difficulties and (partial) solution.. Standard Dimension Reduction algorithms, like SVD [9], need to analyze *all* the input vectors for constructing the distance-preserving matrix M. Namely, the vectors in all the R_o, $o \in O$, matrices, have to be examined. This is impossible here given the distributed setting and the excessive amount of data. To overcome this, C2F employs a recently developed algorithm called the Fast Johnson-Lindenstrauss Transform (FJLT) [4] that randomly generates a matrix M that is guaranteed, with high probability, to have the desired properties. The crux is that the matrix is generated *independently of the input vectors* and no access to the actual data is required. It can thus be created by an arbitrary site and disseminated to all others. Moreover, the FJLT matrix is extremely sparse (hence highly compressible), so the transmission incurs only little communication overhead.

Some difficulties nevertheless still exist. First, it is important to note that FJLT requires the number of original dimensions to be smaller than the number of vectors. In actual systems, however, the number of users always exceeds the number of items, thus FJLT is applicable to the case of shared items but not to shared users.

Second, observe that, when multiplying matrices, the value of all entries must be defined. Since the rating matrices are sparse, some default value (i.e. the average user/item rating) is used for the missing entries [10]. This is common practice when employing similarity measures such as Cosine or L^i distance, but is precisely what Pearson's Correlation Coefficient tries to avoid. (Recall that formula 2 ignores users for which one of the $r_{u,i}$ or $r_{u,j}$ value is missing.). Thus the technique is applicable to the case of Cosine- or L^i-based similarity measures but not for Pearson.

3.2 Dimension Selection

The second algorithm employed by C2F is applicable for both shared items and shared users scenarios and can also handle Pearson. This however comes at the expense of the theoretical worse-case guaranties on the distance error. But we will see that in practice the error turns out nevertheless to be very marginal.

Given an item (user) vector, rather than mapping it to another lower-dimension domain, the algorithm chooses a subset of the original dimensions that best describe the given item (user). Namely, given an item i (user u), the algorithm chooses a small subset of the users (items) such that the similarity of i (u) to other items (users), based solely on this subset of users (items), is close to the "real" similarity value when considering the full users (items) set. We can now send to the collaborating sites a small vector consisting only of the ratings of this reduced set of users (items). The key challenge here is to choose the best set of users/items (one that makes the similarity values as close as possible to the original ones).

To illustrate the idea, let us consider the shared-user scenario. (The shared-items scenario is symmetric). To make things formal, we use the following notation: Recall that in standard (and distributed) CF, the similarity $s_{i,j}$ between two items $i, j \in I$ reflects the correlation between the ratings for these items, given by the full set of users U. Let $U_k \subseteq U$ denote a set of users, of size k, belonging to all the collaborating organizations. We use $s_{i,j}^{U_k}$ to denote the similarity of items i and j when computed based solely on the ratings of users in U_k.

Given an item i and some number k, our goal is to choose a set U_k of users, such that the similarity values $s_{i,j}^{U_k}$, for the items $j \in I$, are as similar as possible to the real similarity values $s_{i,j}$ (computed with the full users set U). As before, we may use Pearson's correlation coefficient to measure the similarity between the two sets of similarity values. We overload notation and, given an item i, use $s_{U,U_k}(i)$ to denote this value.

$$s_{U,U_k}(i) = \frac{\sum_{j \in I}(s_{i,j})(s_{i,j}^{U_k})}{\sqrt{\sum_{j \in I}(s_{i,j})^2}\sqrt{\sum_{j \in I}(s_{i,j}^{U_k})^2}} \quad (4)$$

Note that, unlike in the Pearson equation 2, where the rating had to be normalized (by subtracting the average users rating), the values compared here are unbiased (reflecting the calculated similarity values), hence the simpler shape of the present formula.

Given item i and a number k, we call a set of users U_k, of size k, that has a maximal $s_{U,U_k}(i)$ value, an *optimal k-set*. Unfortunately, an optimal k-set is expensive to find. Clearly, a naive algorithm may simply test all possible k-size sets, namely $O(|U|^k)$ such sets, and choose one with the highest score. Since the number of users may be extremely large, this is too time consuming. Unfortunately we can show that unless P=NP, the exponential dependency in k is unavoidable. Indeed, we can show (by reduction to Set-Cover [2]),

THEOREM 3.1. *The problem of testing, given number k and a bound b, whether there exists a set of k users U_k s.t. $s_{U,U_k}(i) > b$, is NP-complete in k.*

While there are known PTIME approximation algorithms for Set-Cover which could in principle be considered here, our experiments showed that even algorithms that considered just a quadratic (in $|U|$) number of sets were too expensive to be applicable. We have thus chosen to use the following simpler, greedy heuristic: Instead of calculating the similarity value of each possible k-sized subset to U, we consider *singleton* sets, consisting each of a single user. We calculate their similarity to U, then choose for U_k those k users having sets with highest similarity score. [2].

Difficulties and solutions. Some difficulties nevertheless still exist. First, to evaluate equation 4 for $k = 1$ (i.e. a set $U_1 = \{u\}$, consisting of a single user u), the corresponding items similarity scores, $s_{i,j}^{\{u\}}$, need to be computed. Note however that not all similarity measures behave well when only one user is considered. For instance, given a set of users consisting only of one user u, Cosine and Pearson both yield a constant similarity value of 1, for all item pairs i, j rated by u, regardless of the actual rating values (see equation 2) We must thus use a more adequate similarity measure for the single-user sets, e.g. L^i. Note however that if a different measure, e.g. Cosine or Pearson, was employed for full set of users U, we have to make the two similarly values comparable before we can evaluate formula 4. For example, to compare an L^1 similarity with a Pearson one, we need to map the similarity values of the former to the range $[-1, 1]$ of the later (with 1 being the best score and -1 worse). Namely,

$$s_{i,j}^{\{u\}} = 1 - \frac{2\ L^1(i,j)}{MaxRating - MinRating} \quad (5)$$

where $L^1(i,j) = |r_{u,i} - r_{u,j}|$.

A second difficulty is *overfitting* [15]. Intuitively, we would like the selected set of users U_k to "cover" as many items as possible (so that we can use their ranking for the items to determine similarity). However, according to formula 4, a set U_k that ranked only very few items may still get high score if the similarity between these few items, as reflected by these ratings, happens to be the same as their overall similarity value. Following common practice, we solve this problem by applying a *compensation function* to the computed similarity measure, that reflects how many items, out of the relevant ones, are covered by U_k. Namely,

[2]This set may not be unique in the case of multiple users with identical scores, in which case we chose one such set arbitrarily

$$s'_{U,U_k}(i) = s_{U,U_k}(i) \circ \frac{|\ Items(U_k,i)\ |}{|\ Items(U,i)\ |} \quad (6)$$

where $Items(U_k, i)$ (resp. $Items(U, i)$) is the set of items j s.t. U_k (resp. U) contains some user u that rated both i and j. (Recall from Equation 2 that only such items are useful for the similarity computation).

3.3 Signatures Update

We conclude this section by considering updates. The rating matrix gets updated, e.g. when new user ratings are added/updated or when a new user/item is added or deleted. To stay in sync, the relevant signatures need to be correspondingly updated.

For signatures computed via Dimension Selection, updates have a *local* effect, in the sense that the only signatures affected by the update are those of the *site where the update had occurred*. Things are more complex for Dimension Reduction. Some updates here also have local affect. For instance, when a new user is added, or when the ratings of an existing user are updated, the new signature of the user is computed by simply multiplying its new/updated vector by the reduction matrix M. In contrast, the addition of new items have a more *global* affect. Note that the addition of a new item here changes the dimension of the user vectors. To account for this, a new reduction matrix has to be generated (with corresponding dimensions) and disseminated to the collaborating sites (to recompute their signatures accordingly).

The re-computation and dissemination of signatures/reduction matrix with each update, may naturally cause a significant overhand. To avoid this we take lazy approach and update/diseminate the signatures/matrix only periodically, when sufficiently large number of updates had been accumulate. (In between, the old signatures may still be used for estimating the similarity between the corresponding vectors.) For that, we periodically prob locally some of the updated users/items. We compute their accurate new signatures and compare the respective distance that they entails to that obtained using the old signatures. Only when the error exceeds a given threshold, a full re-computation and dissemination is triggered.

4. EXPERIMENTAL EVALUATION

The C2F system allows distinct organizations to collaborate for improving the quality of the recommendations that they provide to users. The system is implemented in Java and designed to be deployed on any CF system implementing its local interface. The experiments were performed on an Intel quad-core machine (Q9400) with 2.66 GHz CPUs, 4GB memory and Windows XP x64 edition.

Acquiring adequate real-life data for the experiments was non trivial. Although several public data sets are available online (e.g. Netflix, Delicious, etc.), which in principle could be ideal for simulating an inter-domain collaboration, the user ids in these data sets are masked, thus common users cannot be identified. We have thus decided to use instead a single data set, and split it into several seemingly uncorrelated subsets. We used the Netflix public data set [6] which provides over 100 millions different ratings of movies by 500,000 users, to 18,000 different movies. To simulate a multi-domain scenario we split the data (using information gathered from the IMDB database [1]) into five seemingly uncorrelated sub-domains (kids movies, documentary, thrillers, etc.) each representing the database of one organization "specializing" in the given area. To compare this to a setting where organization in similar domains are collaborating, we had also considered alterative random splits of the data, once by user and once by movie.

The Netflix data has two parts, a public data set (called the *training set*) and a private set (called *quiz set*) of unreviewed ratings.

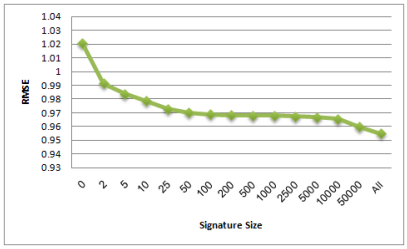

Figure 1: Dimension Selection (multi-domain)

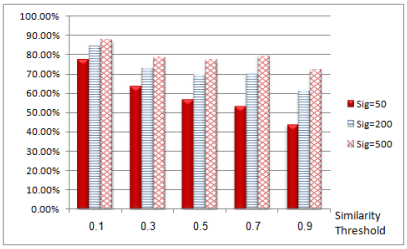

Figure 2: Neighborhood Precision

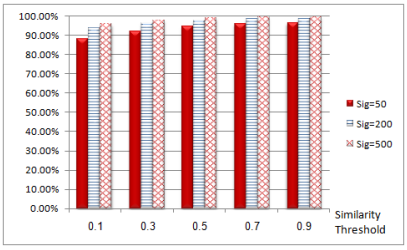

Figure 3: Neighborhood Recall

The quality of a recommender system is measured by comparing its predictions, for the unrevealed ratings in the quiz set, to their true value. The Root Mean Squared Error (RMSE) metric is used to measure accuracy. A lower RMSE value means better results. We note that even a small improvement in the RMSE value (e.g., ~ 0.01) translates to a significant improvement in the quality of the top-k recommended products [12].

Recall, from the Introduction, that our aim here is not to invent yet another CF algorithm, but rather to present a generic technique that allows one to better exploit existing CF algorithms in a distributed, multi-organization setting. Specifically, we experimented with the basic CF algorithm from section 2.1 as well as with more sophisticated variants such as [12]. Our algorithms provided, in all cases, similar relative improvement. We show below a representative sample of the results for the CF algorithm of Section 2.1.

We first consider the Item-based shared-users setting, then User-based shared-items one.

4.1 Item-based CF with shared-users

Recall from section 3 that dimension reduction is not applicable here and may only apply dimension selection. We start with the multi-domain case where movies are split into seemingly uncorrelated organizations.

Figure 1 depicts the RMSE values for increasing signature sizes (number of selected users), in logarithmic scale. The left-most end, when the signature size is 0, represents the classical case where each organization computes its predictions *locally*, with *no information sharing*. At the right-most size, the full item vectors are exchanged. It is evident that collaboration here yields better rating prediction (a drop from RMSE of 1.0207 to 0.9548). We can furthermore see that even the exchange of very small signatures, of size just 50, yields already a significant reduction to 0.97 (a further improvement would requires signatures of size over $50,000$).

To better understand how the size of the signature affects the computation, we examined the item neighborhoods determined by the signatures. As the base for comparison, we consider the neighborhoods determined by the full item vectors. Figures 2 and 3 show the precision (the number of common items divided by the size of the neighborhood determined by the signature) and recall (the number of common items divided by the size of the neighborhood determined by the full item vectors), for increasing threshold values (i.e. smaller neighborhoods) and signatures of size 50, 200 and 500.

Larger signatures naturally yield better precision, as they provide more information about the item whose neighborhood is computed. We can also observe some decrease of precision as the threshold value increases. This is because high threshold implies a smaller neighborhood, which is effected more by even small differences of the similarity values. Note however that that the recall is generally very high, (i.e. the neighborhood contains a large percent of the "real" neighbors) and even increases as the threshold increases (as less "noise" is included in the neighborhood). We note that the results depicted in Figure 1 were obtained with neighborhood threshold of 0.9. But our experiments with smaller neighborhood thresholds yielded similar results: This is because the effect of the noise introduced by smaller thresholds is practically eliminated as it is weighted (see Equation 1) according to its (low) similarity value.

Finally, let us consider performance. Recall that rank prediction in classical centralized CF consists of three main tasks: (1) Computing the similarity between items pairs, (2) determining the items neighborhood based of these values, and (3) consequently computing the rank predictions. In the case of C2F , tasks (2) and (3) remain exactly the same, while task (1) splits into two sub-tasks: (1') signature computation and (1") computing similarity using the (significantly smaller) signatures. Interestingly, our experiments show that the overhead of signature computation (task 1') is compensated by the reduction in the similarity computation time, due to the small size of the compared signatures. This yields an improvement of the overall performance. Figure 4 shows the overall computation time for step 1 (for all item pairs i,j), for varying signature sizes, (compared to the time task 1 takes with the full item vectors). We note that once the similarity values are computed, steps 2 and 3 take just 5 seconds (for the whole data set - 1.4M predictions). So the overall saving, e.g. when using signatures of size 50, is over 50%.

To consider a setting where organizations in *similar* domains are collaborating, we re-run the above experiments with an alterative random splits of the movies into five subsets. The results were similar to what have seen above, showing that our techniques are applicable to both inter- and intra-domain collaboration.

4.2 User-based CF with Shared items

Here one can use both dimension reduction and dimension selection. Before describing our results, recall that in user-based CF, one needs to compute similarities between pairs of users. We first note that due to the huge number of users in the Netflix database, com-

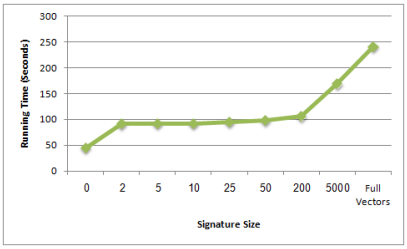

Figure 4: Running times for Dimension selection

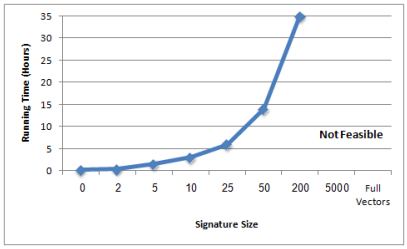

Figure 5: Running times for Dimension reduction

puting similarity values for all user pairs is a very intensive task. Specifically, the number of pairs is too big to store all similarity values in memory, and requires storing them on disk (yielding significant I/O overhead). The use of small signatures, (and the shorter time it takes to computed similarity based on such signatures), becomes essential here. In fact, all our attempts to compute similarities with the full user vectors failed, as the computation never terminated. In contrast, things become feasible with small signatures, even when taking into consideration the extra time it takes to generate them. To illustrate, Figure 5 shows the over all computation time with signatures computed using dimension reduction. (Users are randomly split here into five organizations). The results for dimension selection were similar.

In terms of recommendations quality, we observed improvements similar to what we have seen in Figure 1, for the Item-based case, (up to signatures of size 200, after which the computation was no longer feasible here). Interestingly, the improvement for signatures generated via dimension selection was slightly better than that of dimension reduction(0.97 vs. 0.98). This is because dimension selection allows using the more accurate Pearson similarity measure (see previous section).

To conclude, recall that the dimension reduction algorithm provides a worse case bound on the error in neighborhood values, whereas dimension reduction does not. In practice however the precision and recall values (for the computed neighborhoods) that both algorithm exhibit are similar, and behave as what we have seen above for the Item-based setting.

4.3 Updates

To evaluate the sensitivity of signatures to updates, we started with partial ranking matrix consisting overall of 100K users (20% of the full data set). Then, we gradually added the remaining rankings. At each point we compared the quality of the predictions obtained with the signatures computed based on the initial data set, to the one obtained with current signatures. We repeated the experiment for signatures computed via dimension selection and reduction and of varying sizes. Our experiments consistently showed that the increase in RMSE value was fairly small - just 0.001 for 100K added users. This demonstrate the adequacy of our lazy update propagation policy, and follows from the typical fairly consistent distribution of user opinions w.r.t. the large number items.

5. RELATED WORK AND CONCLUSION

We considered in this paper a popular class of recommender systems, based on Collaborative Filtering. We argued that a *multi-organization* collaboration, even for organizations operating in different subject domains, can greatly improve the quality of the recommendations that the individual organizations provide to their users. To substantiate this claim, we presented C2F , a system that employs CF in a distributed fashion and improves the quality of the generated recommendations, while reducing the amount of data exchanged between the collaborating parties.

Extending our approach to *content-based* systems that also use information about item semantics is a challenging future research.

Previous work on CF in a distributed setting focused on P2P architecture, typically aiming to speed up the computation. A common solution is to decentralize the P2P network w.r.t the users (items), maintaining a "buddies" table at each pier, pointing to the closest users (items) which are believed to share the same taste (be similar) [8, 17, 16]. Another architecture is presented [5] where the authors describe the recommendation mechanism of the popular TiVo: The network here consists of a centralized server which first accumulates the ratings from all the devices periodically, and then evaluated the similarity between all items (shows). In all these works the network architecture differs fundamentally from our setting: they consider network of thousands computers, each holding an assigned small part of the data, (useless by itself), whereas we target a much smaller set of collaborating organizations (servers), each holding an entire data set from its corresponding domain.

Closest to our work is [7] that considers the aggregation of rate predictions from multiple sites. The focus however on similar domains with common (user,item) pairs, whereas our solution allows for *inter-domain* collaboration, exchanging only a small amount of information between the collaborating parties, yet sufficient for effectively identifying and exploiting the relevant information.

A complementary line of research considers privacy. For instance, [14] the authors show how to preserve the privacy of the users by aggregating the data (ratings) among several users, thus eliminating the identification of the ratings, and then uploading the combined information to the server. We note that our dimension reduction algorithm also aggregates user ratings (the multiplication with the matrix M transform the individual user ratings into linear combination of such ratings) and we intend to study its implication to privacy in future research.

6. REFERENCES

[1] Imdb interface. *http://www.imdb.com/interfaces/*.

[2] Set-cover problem. *http://en.wikipedia.org/wiki/Set_cover_problem*.

[3] G. Adomavicius and A. Tuzhilin. Towards the next generation of recommender systems. *IEEE TKDE*, 2005.

[4] N. Ailon and B. Chazelle. Faster dimension reduction. *Communications of the ACM*, 2010.

[5] K. Ali and W. van Stam. Tivo: Making show recommendations using a distributed collaborative filtering architecture. *KDD*, 2004.

[6] J. Bennet and S. Lanning. The netflix prize. *KDD Cup*, 2007.

[7] S. Berkvosky, T. Kuflik, and F. Ricci. Distributed collaborative filtering with domain specialization. *RecSys*, 2007.

[8] A. E. H. Byeong Man Kim, Qing Li. A decentralized cf approach based on cooperative agents. *WWW*, 2006.

[9] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JASIST*, 1990.

[10] I. Fodor. A survey of dimension reduction techniques. *Communications of the ACM*, 2002.

[11] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *JMLR*, 2003.

[12] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. *KDD*, 2008.

[13] J. L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 1988.

[14] R. Shokri, P. Pedarsani, G. Theodorakopoulos, and J.-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. *RecSys*, 2009.

[15] T. I. V., L. D. J, and A. I. Luik. Comparison of overfitting and overtraining. *American Chemical Society*, 1995.

[16] J. Wang, J. Pouwelse, R. L. Lagendijk, and M. J. Reinders. Distributed collaborative filtering for peer-to-peer file sharing systems. *SIGIR*, 2005.

[17] J. Wang, M. J. Reinders, R. L. Lagendijk, and J. Pouwelse. Self-organizing distributed collaborative filtering. *SIGIR*, 2005.