Collaborative Filtering Process in a Whole New Light

Panagiotis Symeonidis Alexandros Nanopoulos Apostolos Papadopoulos
Yannis Manolopoulos
Aristotle University, Department of Informatics, Thessaloniki 54124, Greece
{symeon, alex, apostol, manolopo}@delab.csd.auth.gr

Abstract

Collaborative Filtering (CF) Systems are gaining widespread acceptance in recommender systems and e-commerce applications. These systems combine information retrieval and data mining techniques to provide recommendations for products, based on suggestions of users with similar preferences. Nearest-neighbor CF process is influenced by several factors, which were not examined carefully in past work. In this paper, we bring to surface these factors in order to identify existing false beliefs. Moreover, by being able to view the "big picture" from the CF process, we propose new approaches that substantially improve the performance of CF algorithms. For instance, we obtain more than 40% percent increase in precision in comparison to widely-used CF algorithms. We perform an extensive experimental evaluation, with several real data sets, and produce results that invalidate some existing beliefs and illustrate the superiority of the proposed extensions.

Keywords: Recommender Systems, Collaborative Filtering, Information Retrieval

1 Introduction

Information Filtering has become a necessary technology to attack the "information overload" problem. In our everyday experience, while searching on a topic (e.g., products, movies, etc.), we often rely on suggestions from others, more experienced on it. In the Web, however, the plethora of available suggestions renders it difficult to detect the trustworthy ones. The solution is to shift from individual to collective suggestions. Collaborative Filtering (CF) applies information retrieval and data mining techniques to provide recommendations based on suggestions of users with similar preferences. CF is a very popular method in recommender systems and e-commerce applications. Two types of CF algorithms have been proposed: (a) *nearest-neighbor* (a.k.a. memory-based) algorithms, which rely on finding the most similar ones among the past users, and (b) *model-based* algorithms, which develop a model about user ratings. Research results and practical experience have reported that nearest-neighbor algorithms present good performance in terms of accuracy, for multi-value rating data [5].

Nearest-neighbors CF algorithms are influenced by several factors. Related research on CF, during the past decade, approached some of these factors. However, existing approaches may not be considered complete, because they examine the various factors only partially. More specifically, existing CF algorithms and their experimental evaluation focus only on parts of the CF process and do not handle it as a whole. For the aspects that these partial considerations do not examine, they usually make choices, which our study demonstrates that can be misleading. Through our study we are also able to confirm that there exist dependencies between the factors affecting CF. Therefore, we have to perform an evaluation of the entire CF process in order to produce reliable conclusions.

In this work, we focus on nearest-neighbor CF algorithms. First, we provide a thorough analysis of the factors involved in CF. During the analysis we identify choices that have been incorrectly adopted and new issues that have not been considered so far. As a result, we propose several extensions and new approaches, which substantially improve the entire CF process. We have carried out extensive experimental evaluation, which takes into account many new aspects. Our results: (i) reveal fallacies in existing beliefs about CF algorithms, (ii) demonstrate the superiority of the proposed methods (more than 40% improvements in terms of precision over widely-used CF algorithms), (iii) provide many insights about further examinations in future work.

The rest of this paper is organized as follows. Section 2 summarizes the related work, whereas Section 3 contains the analysis of the examined CF factors. The proposed approach is described in Section 4. Experimental results are given in Section 5. Finally, Section 6 concludes this paper.

2 Related work

In 1992, the Tapestry system [1] introduced Collaborative Filtering (CF). In 1994, the GroupLens system [9] implemented a CF algorithm based on common users preferences.

Nowadays, it is known as user-based CF (UB) algorithm, because it employs users' similarities for the formation of the neighborhood of nearest users. Since then, many improvements of user-based algorithm have been suggested, e.g., [3].

In 2001, another CF algorithm was proposed. It is based on the items' similarities for a neighborhood generation [11, 6]. Now, it is denoted as item-based or item-item CF (IB) algorithm, because it employs items' similarities for the formation of the neighborhood of nearest users.

Most recent work followed the two aforementioned directions (i.e., user-based and item-based). Herlocker et al. [4] weight similarities by the number of common ratings between users/items, when it is less than some threshold parameter γ. Xue et al. [13] suggest a hybrid integration of aforementioned algorithms with model-based CF algorithms. In the following section we elaborate further on related work, through the analysis of the factors we examine.

3 Factors affecting the CF process

In this section, we identify the major factors that critically affect all CF algorithms. Our analysis focuses on the basic operations of the CF process, which consists of three stages.

- Stage 1: formation of user or item neighborhood.
- Stage 2: top-N list generation algorithms.
- Stage 3: assessment of the recommendation list.

In the rest of this section we elaborate on the aforementioned factors, which are organized with respect to the stage that each one is involved. Table 1 summarizes the symbols that are used in the sequel.

Symbol	Definition
k	number of nearest neighbors
N	size of recommendation list
P_τ	threshold for positive ratings
I	domain of all items
U	domain of all users
u, v	some users
i, j	some items
I_u	set of items rated by user u
U_i	set of users rated item i
$r_{u,i}$	the rating of user u on item i
$\overline{r}_u$	mean rating value for user u
$\overline{r}_i$	mean rating value for item i
$p_{u,i}$	predicted rate for user u on item i
γ	threshold value used by WS

Table 1. Symbols and definitions.

3.1 First stage factors

Amount of sparsity: In many real cases, users rate only a very small percentage of items, thus rating data become sparse. Due to lack of sufficient information, sparsity leads to inaccurate recommendations. For this reason, several recent works concentrate only on sparse data [4, 6, 11] (e.g., Movielens). On the other hand , however, there exist dense rating data sets (e.g., Jester [2]).

Train/Test data size: There is a clear dependence between the train set's size and the accuracy of CF algorithms [11]. Through our experimental study we verified this conclusion. Additionally, we saw that after an upper threshold of the train-set size, the increase in accuracy is less steep. However, the effect of overfitting is less significant compared to general classification problems. In contrast, low train set sizes negatively impact accuracy. Therefore, the fair evaluation of CF algorithms should be based on adequately large train sets. Though most related research uses a size around 80%, there exist works that use significantly smaller sizes [8]. From our experimental results we concluded that an 75% train-set size corresponds to an adequate choice. But we have to notice that training/test size should not be data set independent.

Neighborhood size: The number, k, of nearest neighbors, used for the neighborhood formation, directly affects accuracy. Related work [3, 10] utilizes a k in the range of values between 10 and 100. The optimum k depends on the data characteristics. Therefore, CF algorithms should be evaluated against varying k. Moreover, an issue that has not been precisely clarified in related work, is whether we include in the neighborhood a user or item with negative similarity. In order to improve accuracy, we suggest keeping only the positive similarities for the neighborhood formation, even if less than the specified number k of neighbors remain.

Similarity measure: The most extensively used similarity measures are based on correlation and cosine-similarity [4, 8, 11]. Specifically, user-based CF algorithms mainly use Pearson's Correlation (Equation 1), whereas for item-based CF algorithms, the Adjusted Cosine Measure is preferred (Equation 2) [8, 11]. The Adjusted Cosine Measure is a variation of the simple cosine formula, that normalizes bias from subjective ratings of different users.

We examined all measures that have been discussed in related work and noticed that the use of different measures did not result in substantial differences in performance. As default options, for user-based CF we use the Pearson Correlation, whereas for item-based we use the Adjusted Cosine Similarity, because they presented the best behavior.

$$\mathrm{sim}(u,v) = \frac{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)(r_{v,i} - \overline{r}_v)}{\sqrt{\sum_{\forall i \in S} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall i \in S} (r_{v,i} - \overline{r}_v)^2}}, S = I_u \cap I_v. \quad (1)$$

$$\mathrm{sim}(i,j) = \frac{\sum_{\forall u \in T} (r_{u,i} - \overline{r}_u)(r_{u,j} - \overline{r}_u)}{\sqrt{\sum_{\forall u \in U_i} (r_{u,i} - \overline{r}_u)^2} \sqrt{\sum_{\forall u \in U_j} (r_{u,j} - \overline{r}_u)^2}}, T = U_i \cap U_j. \quad (2)$$

Herlocker et al. [4] proposed a variation of the aforementioned measures, which weights similarities by the number

of common item ratings when it is less than some threshold parameter γ. Henceforth, this measure is denoted as Weighted Similarity (WS). If *sim* is the similarity measure computed either with Equation 1 or Equation 2, then WS is equal to $\frac{\max(c,\gamma)}{\gamma} sim$, where c is the number of co-rated items.

Pearson's Correlation (Equation 1) takes into account only the set of items, S, that are *co-rated* by both users. Moreover, in Equation 1 mean ratings $(\overline{r}_v, \overline{r}_u)$ in numerator are computed only from co-rated items. Similarly, the Adjusted Cosine Measure (Equation 2) considers in the numerator only the set of users, T, that co-rated both the examined pair of items, whereas means are taken over all ratings for a user, not a subset of ratings shared with any other user. In contrast, the denominator of Equation 2 does not consider only the users that co-rated both the items. It is worth noticing that the clear definition of sets S or T for co-rated items only, is an issue that is implied in all related work. However, it is unfortunate that existing work expresses Equations 1 and 2 in very different forms, whereas only few of them [11, 8] are precisely clear on the aforementioned definition for S and T. Based on this observation and on the different use of co-rated items in the denominators of Equation 1 and 2, we wanted to research different definitions for sets S and T.

3.2 Second stage factors

Recommendation list's size: The size, N, of the recommendation list corresponds to a tradeoff: With increasing N, the absolute number of relevant items (i.e., recall) is expected to increase, but their ratio to the total size of the recommendation list (i.e., precision) is expected to decrease. (Recall and precision metrics are detailed in the following.) In related work [6, 11], N usually takes values between 10 and 50.

Generation of recommendation list: The most often used technique for the generation of the top-N list, is the one that counts the frequency of each item inside the found neighborhood, and recommends the N most frequent ones [10]. Henceforth, this technique is denoted as Most-Frequent item recommendation (MF). MF can be applied to both user-based and item-based CF algorithms. Karypis [6] reports another technique, which additionally considers the degree of similarity between items. This takes into account that the similarities of the k neighbors may vary significantly. Thus, for each item in the neighborhood, this technique admeasures not just their number of appearances, but the similarity of neighbors as well. The N items with the highest sum of similarities are finally recommended. Henceforth, this technique is denoted as Highest-Sum-of-Similarities item recommendation (HSS). HSS is applicable only to item-based CF.

Based on the aforementioned rationale, we wanted to perform an examination of other additional criteria against MF (used in the majority of existing works), in order to examine if this direction is promising for future work, because besides [6], very few works elaborate on this issue.

Positive rating threshold: It is evident that recommendations should be "positive". It is not success to recommend an item that will be rated with 1 in scale 1-5. Nevertheless, this issue is not clearly defined in several existing works. We argue that "negatively" rated items should not contribute to the increase of accuracy, and we use a rating-threshold, P_τ, to recommended items whose rating is no less than this value. If not a P_τ value is used, then the results can become misleading, since negative ratings can contribute to the measurement of accuracy.

3.3 Third stage factors

Evaluation Metrics: Several metrics have been used for the evaluation of CF algorithms, for instance the Mean Absolute Error (MAE) or the Receiving Operating Characteristic (ROC) curve [4, 5]. MAE represents the absolute differences between the real and the predicted values and is an extensively used metric. From our experimental study (Section 5) we understood that MAE is able to characterize the accuracy of prediction, but is not indicative for the accuracy of recommendation. Since in real-world recommender systems the experience of users mainly depends on the accuracy of recommendation, MAE may not be the preferred measure. For this reason we focus on widely accepted metrics from information retrieval. For a test user that receives a top-N recommendation list, let R denote the number of *relevant recommended items* (the items of the top-N list that are rated higher than P_τ by the test user). We define the following:

- *Precision* is the ratio of R to N.
- *Recall* is the ratio of R to the total number of relevant items for the test user (all items rated higher than P_τ by him).

Notice that with the previous definitions, when an item in the top-N list is not rated at all by the test user, we consider it as *irrelevant* and it counts negatively to precision (as we divide by N) [8]. In the following we also use $\mathrm{F}_1 = 2 \cdot \mathrm{recall} \cdot \mathrm{precision} / (\mathrm{recall} + \mathrm{precision})$. F_1 is used because it combines both the previous metrics.

Setting a baseline method: Existing experimental evaluations lack the comparison against a baseline algorithm. A baseline algorithm has to be simple and to indicate what can be attained with as little effort as possible. Through a baseline, we can see the actual improvement due to existing CF algorithms.

Past/future data: In real-world applications, recommendations are derived only from the currently available ratings of the target user. However, in most of related works the entire set of rated items is considered apriori known. For a more realistic evaluation, recommendations should consider

the division of items into two sets: (i) the past items and (ii) the future items. To make it more clear, lets see an example. Assume that a test user has rated 10 items. According to the previous work, these ratings are used to calculate his similarity with the train users. In contrast, we want to measure how algorithms react when we use a smaller number of items than those he has rated. For instance, if we use 2 items(past items) from the 10 can we find the other 8(future items). If only a fraction of items is included in the past set, then the accuracy is expected to decrease compared to the best case. For this reason, we study the effect of this issue.

4 Proposed methodology

In the sequel, we describe in detail our proposed extensions. These extensions are based on the "big picture" of the CF process.

4.1 Extensions for the first stage : the UNION similarity measures

We first examined the two most important factors that are involved in the first stage: sparsity and the similarity measure. As mentioned, for the formation of sets S and T (see Equations 1 and 2), past work [5, 8, 11] takes into account only the items that are co-rated by both users. For instance, Figure 1 depicts the ratings of two users, U_1 and U_2, over five items. When only co-rated items are considered, then the similarity measure will be computed based on the ratings for I_1 and I_3.

	I_1	I_2	I_3	I_4	I_5
U_1	3	-	5	4	-
U_2	4	2	4	-	-

Figure 1. Example of the ratings of two users over four items (dash denotes an empty value).

In case of sparse data, by constraining S and T with co-rated items, we reduce further the effective amount of used information. For sparse data, the rated items correspond to a very small percentage. However, when S and T are formed by co-rated items only, we additionally ignore many of them, that is, S and T will finally include an even smaller percentage of items. To avoid this, we considered [12] another definition for S and T:

$$S = I_u \cup I_v, \quad T = U_i \cup U_j \tag{3}$$

Thus, S includes items for which at least one of the users rates an item. In the example of Figure 1, except the ratings for I_1 and I_3, the ratings for I_2 and I_4 will be considered too. Similar reasoning is followed for the set T for the case of item-based CF.

By combining the definitions of S and T given in Equation 3 with the Pearson correlation and adjusted cosine similarity measures, we get two reformed measures: UNION Pearson correlation (for UB) and UNION adjusted cosine (for IB), respectively.[1] Notice that in case of UNION Pearson correlation, user mean ratings correspond to the average user ratings over all rated items. Moreover, in Equation 2, the arithmetic means are divided with all rated items. This practically means that if a data set is sparse, then the arithmetic means tend to have close to zero values, thus the measure essentially becomes equivalent to that of pure cosine.

With the extended definitions of S and T, more information is being exploited. For instance, assume that a user rates an item with 5 (in 1–5 scale) and another one does not rate this item at all. This fact denotes a clear difference in their preference for this item, which can be exploited when few information is provided, as it is the case for sparse data. By taking the existing definitions of S and T, this information is being ignored, because it has no impact on the computation of the similarity value. In contrast, by taking the extended definitions, this fact is taken into account.

4.2 Extensions for the second stage: the HPR and HSR algorithms

In Section 3.2, we described the two criteria, MF and HSS, that have been proposed so far for the generation of the top-N list. The ranking criteria are important, as they can significantly affect the performance of CF. For this reason we examined additional criteria, to see if this direction of research worths investigation.

As a first extension of the existing ranking criteria, someone could use the predicted values for each item to rank them. Predicted values [8] are computed by Equations 4 and 5, for the cases of user-based and item-based CF, respectively. These equations have been used in related work for the purpose of MAE calculation, whereas we use them for generation of top-N recommendation list.

$$p_{u,i} = \overline{r}_u + \frac{\sum_{v \in U} \text{sim}(u,v)(r_{v,i} - \overline{r}_v)}{\sum_{v \in U} |\text{sim}(u,v)|} \tag{4}$$

$$p_{u,i} = \overline{r}_i + \frac{\sum_{j \in I} \text{sim}(i,j)(r_{u,j} - \overline{r}_j)}{\sum_{j \in I} |\text{sim}(i,j)|} \tag{5}$$

Therefore, we sort (in descending order) the items according to predicted rating value, and recommend the first N of them.[2] This ranking criterion, denoted as Highest Predicted Rated item recommendation (HPR) is influenced by the good accuracy of prediction that existing related work reports through the MAE. HPR opts for recommending the items

[1] Henceforth, when it is clearly understood from the context whether we discuss about UB or IB, we use only the name UNION.

[2] If less than N items have positive ratings (i.e., not less than P_τ), then less than N items remain in the list.

that are more probable to receive a higher rating. Nevertheless, as already mentioned, MAE is not indicative for the accuracy of recommendation. As our experimental results will demonstrate, HPR presents poor performance. This fact is another indication that MAE alone cannot characterize the performance of CF algorithms.

As another criterion, which resulted from observations during our experimental investigation, we sum the positive rates of the items in the neighborhood, instead of just counting their frequency. This criterion is denoted as Highest Sum of Rates item recommendation (HSR). The top-N recommendation list consists of the N items with the highest sum. The intuition behind HSR is that it takes into account both the frequency (as MF) and the actual ratings, because it wants to favor items that appear most frequently in the neighborhood *and* have the best ratings. Assume, for example, an item i that has just a smaller frequency from an item j. If i is rated much higher than j, then HSR will prefer it from i, whereas MF will favor j.

4.3 Extensions for the third stage : the Baseline algorithm

Considering the factors described in Section 3.3 regarding the evaluation procedure, we detail a baseline algorithm. We propose the one that recommends the N items that are most frequently rated positively in the entire training set. This algorithm is denoted as BL. BL is very simple and, as will be shown in our experimental results, it is quite effective. For instance, our experiments with Movielens-100K data set have shown that, with BL, when we simply propose the N = 20 most positively rated movies (20 most popular movies), precision reaches 40%. Therefore, the most frequently rated items are very probable to be selected by the majority of the users. For the aforementioned reasons, BL is a tool to clearly evaluate the actual improvement of existing CF algorithms. We will see in our experiments that asymptotically, as k (neighborhood size) increases, the performance of Pearson correlation and adjusted cosine tend to become equal with that of BL. In the extreme case where k is equal to the number of all users in the training set, the result of the Most-Frequent item-recommendation procedure, for the generation of the top-N list, becomes equivalent to BL.

5 Performance study

In the sequel, we study the performance of the described extensions against existing CF algorithms, by means of a thorough experimental evaluation. Both user-based and IB algorithms are tested. Several factors are considered, like the sparsity, the similarity measures, and criteria for generating the top-N list. The additional factors, that are treated as parameters, are the following: the neighborhood size (k, default value 10), the size of the recommendation list (N, default value 20), the size of train set (default value 75%), and the division between past/future data. Regarding WS, the γ value was set to 25. Evaluation is performed with the precision and recall metrics (given as percentages). We also use F_1 metric.

We perform experiments with four real data sets that have been used as benchmarks in prior work. In particular, we examined two MovieLens data sets: (i) the first one with 100,000 ratings assigned by 943 users on 1,682 movies, and (ii) the second one with about 1 million ratings for 3,592 movies by 6,040 users. The range of ratings is between 1(bad)-5(excellent) of the numerical scale. Moreover, we ran our experiments on the EachMovie data set [7], which contains 2,811,983 ratings entered by 72,916 for 1628 different movies, and the Jester data set, which contains 4.1 million ratings of 100 jokes from 73,496 users. Due to lack of space, we present results only for the first MovieLens and the Jester data sets (the default set is the former), because they correspond to a sparse and a dense data set, respectively. The performance of the former has been verified with the results of the other two (sparse) real data sets. Finally, in all data sets, we normalized the rating scale in the range 1–5, whereas P_τ is set to 3 and the value of an unrated item is considered equal to zero.

5.1 Results for user-based CF

First, we examine the UB CF case and compare the existing Pearson similarity and WS measures against UNION. We also include the baseline (BL) algorithm. The results for precision and recall vs. k are displayed in Figure 2a and b, respectively.

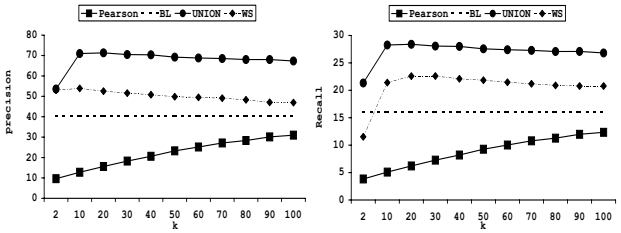
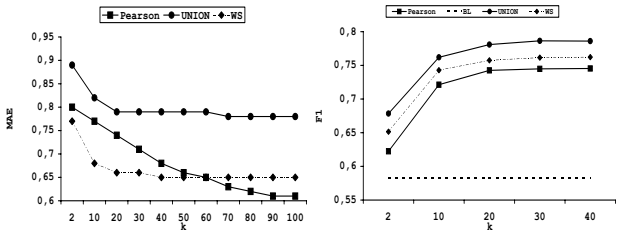


Figure 2. Performance of user-based CF vs. k: (a) precision, (b) recall.

As shown, the existing Pearson measure, which is based on co-rated items, performs worst than BL. This result is surprising, as BL is very simple. WS improves Pearson, because the disadvantage of Pearson, due to co-rated items, is downsized by the weighting with the number of common items. UNION clearly outperforms all other measures for the reason that have been described in Section 4. Outside the examined k range (not displayed), Pearson stabilizes and never exceeds BL. As we already described, with increasing k, Pearson measure practically becomes equivalent to BL.

We now examine the MAE metric. Results are illustrated in Figure 3a (BL is only for recommendation, not prediction, thus omitted). As expected, Pearson yields the lowest MAE values, whereas WS is second best. This fact supports our explanation that MAE is indicative only for the evaluation of prediction and not of recommendation, as these measures did not attain the best performance in terms of precision and recall.

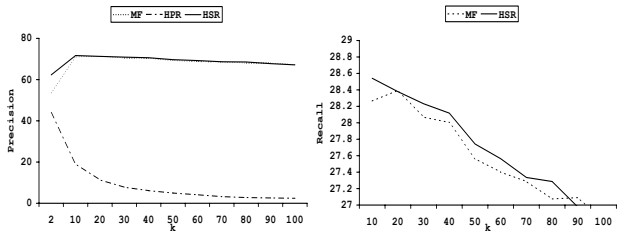


(a) (b)

Figure 3. Performance of user-based CF vs. k: (a) MAE, (b) F_1 for dense data.

To consider the impact of density, we also examine the Jester data set. The results for the F_1 metric are depicted in Figure 3b. In this case, the relative differences are smaller than for the case of sparse data. The reason is that dense data have a sufficient amount of information, thus there is less need to exploit information in the way UNION does. Nevertheless, UNION still presents the best performance.

Finally, we test the described criteria for the top-N list generation: MF, HPR, and HSR. We used the UB UNION similarity measure, because it presented the best performance in the previous measurements. The results for precision and recall are given in Figure 4a and b, respectively.



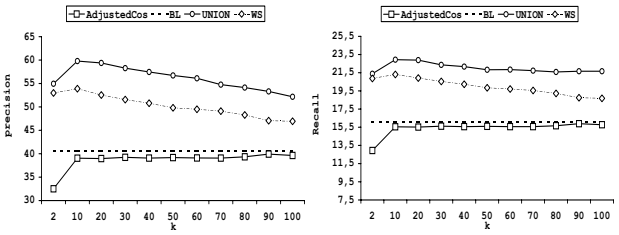
(a) (b)

Figure 4. Comparison of criteria for the generation of top-N list for user-based CF vs. k: (a) precision, (b) recall.

In Figure 4a it is shown that HPR is clearly outperformed by the other two criteria, a fact that furthermore illustrates the unsuitability of the MAE metric to characterize the task of recommendation. On the other hand, MF and HSR present similar precision. Figure 4b presents the results on recall (to make comparison between MF and HSR more clear, we omit HPR). HSR is constantly better than MF, though slightly.

5.2 Results for item-based CF

We perform similar measurements for the case of IB CF. Thus, we first examine the precision and recall for the adjusted cosine (considers co-rated items) against UNION. The results are depicted in Figure 5 and are analogous to those of the UB case. UNION clearly outperforms adjusted cosine and WS. Again, it is surprising to find that the adjusted cosine looses out by BL.



(a) (b)

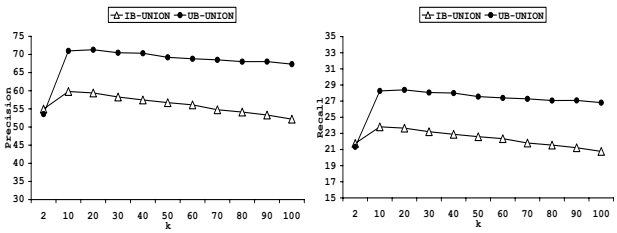
Figure 5. Performance of item-based CF vs. k: (a) precision, (b) recall.

Regarding the examination of the dense data set (Jester), we have to notice that, since IB CF has been designed to suit the needs of sparse data, we find out that for dense data all IB algorithms are outperformed by BL. This is the case even for UNION, although it performs better than adjusted cosine. This result clarifies the need to examine CF algorithms for all the involved factors, in this case the amount of sparsity, in order to draw more complete conclusions. Finally, we test the three described criteria, MF, HPR, and HSS, for the top-N list generation by IB CF algorithms. The results for precision and recall are analogous to the UB case (Due to lack of space we do not present the aforementioned results).

5.3 Comparative results

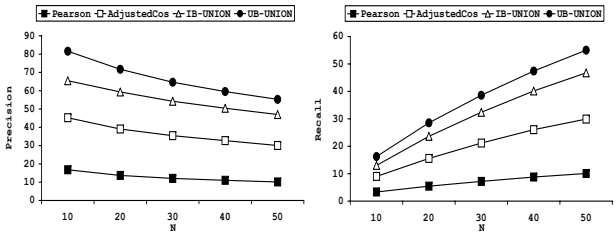
In this section, we compare UNION for the UB and IB cases, as the corresponding UNION measures were shown to have the best performance in each case separately. The results for precision are depicted in Figure 6a, whereas those for recall are depicted in Figure 6b.

These results demonstrate that UB CF compares favorably against IB CF when UNION is used. The difference in precision is larger than 10%, whereas with respect to recall, it exceeds 5% (we refer to the optimum values resulting from the tuning of k). This conclusion contrasts the existing one, that IB is more preferable than UB, for the case of sparse data. The reason is that UB CF is more focused towards the preferences of the target user. In contrast, with IB CF, the recommended items may have been found similar by transactions of users with much different preferences than the ones of the target user. Thus, they may not directly reflect the preferences of the latter. However, this property



(a) (b)

Figure 6. Comparison between UB and IB: (a) precision, (b) recall.



(a) (b)

Figure 7. Comparison vs. N: (a) precision, (b) recall.

could not be revealed with the existing similarity measures and evaluation procedures.

The previous conclusion is in accordance with the one resulting from the comparison for the dense data set (Jester). Due to lack of space we do not present a graph for this case. However, from Figure 3b and Figure **??**b it is easy to see that UB performs much better than IB when UNION is used, as the former is better than BL and the latter is worse.

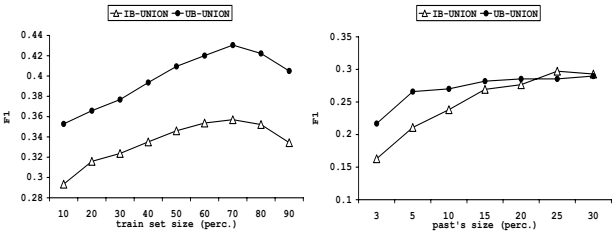
Finally, we have to notice that IB algorithms employ offline computation(items' similarity matrix), which is an advantage over UB algorithms in terms of execution time. Furthermore, the generation of top-N list for the UB approach further burdens the CF process. The reason is that the algorithm finds, firstly, user neighbors in the neighborhood matrix and then counts presences of items in the user-item matrix. In contrast, with the IB approach the whole work is completed in the item neighborhood matrix.

5.4 Examination of additional factors

In this section we examine the impact of the additional factors. In our measurements we consider the existing two cases, that is, UB CF with Pearson similarity and IB CF with Adjusted Cosine.

Recommendation list's size: First, we examine the impact of N (recommendation list's size). The results are depicted in Figure 7. As expected, with increasing N, recall increases and precision decreases. The relative differences between the algorithms are coherent with those in our previous measurements. We have to mentioned that in real applications, N should be kept low, because it is impractical for a user to see all recommendations when their number is large.

Training/Test data size: We test the impact of the size of the train set, which is expressed as percentage of the total data set size. The results for the F_1 metric are given in Figure 8a (to make the graph more clear, we show results only for the two best UB and IB algorithms). As expected, when the train set is small, performance downgrades for both algorithms. Therefore, we should be careful enough when we evaluate CF algorithms and use adequately large train sets. Similar to the previous measurements, in all cases UB UNION is better than IB UNION. The performance of both reaches a peak around 75%, after which it reduces. The reason for this is the overfitting that results from very large train sets. Thus, after a threshold of the training set size, the increase in accuracy for algorithms is less steep. However, the effect of overfitting is less significant compared to general classification problems. In contrast, low training set sizes negatively impact accuracy. Therefore, the fair evaluation of CF algorithms should be based on adequately large training sets.



(a) (b)

Figure 8. (a) Comparison vs. train set size. (b) Comparison vs. past's size.

Past/future data: We examine the performance when considering the division between past and future data. For each user's transaction in the train set we keep the 70% as future data and use a varying number of ratings from the rest 30% as past data. This way, we examine the impact of past's size. We compare UB UNION with IB UNION using the F_1 metric. The results are illustrated in Figure 8b. As the size of the considered past reduces, the performance of both algorithms reduces. This result demonstrates the need to evaluate CF algorithms using the division between past and future data, because this division is more indicative about the actual performance in real-world CF applications. Nevertheless, UB UNION manages to keep a higher F_1 value for small past size, whereas both algorithms converge to the same point for larger past. The merit of UB UNION is evident, as in real applications the past size takes very small values.

6 Conclusions

We have performed a thorough study of neighborhood-based CF, which brought out several factors of the three stages CF process, that have not been examined carefully in the past. Specifically, we brought to surface issues concerning, (i) the formation of user's neighborhood, (ii) the top-N list generation algorithms and, (iii) the quality assessment of the top-N recommendation list. We carried out extensive experimentation which reforms several existing beliefs and provides new insights.

In particular, we highlight the following conclusions from our examination:

- In contrast to what is reported in majority of related work, MAE is not indicative for the accuracy of the recommendation process. It is, though, useful to characterize the quality of the similarity measure (as reflected in the process of prediction).
- Constraining similarity measures with co-rated items, weaknesses the measure. Though it is somewhat useful to consider the number of co-rated items (as WS does), the strict constraining inside the formulae for similarity measures is not suitable.
- The proposed extensions that do not use co-rated items only, substantially improve the performance of CF, especially for sparse data.
- The generation of the top-N recommendation list with a ranking criterion, is a significant problem that worths further consideration. The proposed HSR criterion can replace the existing MF, which has been used by the majority of related work.
- Our results showed that, following the best options for UB and IB CF, the former compares favorably to the latter. This contrasts with existing results in related work, because until now, comparison did not follow the best options we describe.
- Finally, we included a baseline (BL) algorithm, which was useful to better characterize the performance of existing CF algorithms. Actually, the comparison against BL has produced some surprising results.

We have to notice that item-based algorithms employ offline computation, which is an advantage over user-based algorithms in terms of execution time. For this reason, in our future work we will consider the issue of scalability through techniques like SVD, and compare the two approaches for this factor as well. Moreover, we will examine other techniques for the generation of the top-N recommendation list. Finally, we will consider the issue of an approach that would unify the best characteristics of these two cases in an integrated approach.

References

[1] D. Goldberg, D. Nichols, M. Brian, and D. Terry. Using collaborative filtering to weave an information tapestry. *ACM Communications*, 35(12):61–70, 1992.

[2] K. Goldberg, T. Roeder, T. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[3] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of ACM SIGIR Conference*, pages 230–237, 1999.

[4] J. Herlocker, J. Konstan, and J. Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*, 5(4):287–310, 2002.

[5] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.

[6] G. Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of ACM CIKM Conference*, pages 247–254, 2001.

[7] P. McJones and J. DeTreville. Each to each programmer's reference manual. *Tech. Rep. Systems Research Center*, 1997-023.

[8] R. McLauglin and J. Herlocher. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of ACM SIGIR Conference*, pages 329–336, 2004.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering on netnews. In *Proceedings of Computer Supported Collaborative Work Conference*, pages 175–186, 1994.

[10] B. Sarwar, G. Karypis, J. Konstan, and R. J. Analysis of recommendation algorithms for e-commerce. In *Proceedings of ACM Electronic Commerce Conference*, pages 158–167, 2000.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of WWW Conference*, pages 285–295, 2001.

[12] P. Symeonidis, A. Nanopoulos, A. Papadopoulos, and Y. Manolopoulos. Collaborative filtering: Fallacies and new insights in measuring similarity. In *Proceedings on Web Mining Workshop (PKDD conference)*, 2006.

[13] G. Xue, C. Lin, and Q. e. a. Yang. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of ACM SIGIR Conference*, 2005.