Recommender Systems: an Experimental Comparison of two Filtering Algorithms

Emmanouil G. Vozalis and Konstantinos G. Margaritis

Department of Applied Informatics, University of Macedonia
54006 Thessaloniki, Greece

Abstract. In this work we provide a review of the experiments we conducted on two contrasting recommender systems' algorithms: classic Collaborative Filtering and Item-based Filtering. We discuss the results extracted from the experiments and test the validity of the claim that Item-based Filtering improves significantly on the performance of classic Collaborative Filtering. Finally, the results are compared with a smart non-personalized algorithm in order to evaluate the methods' usefulness.

1 Introduction

Recommender Systems were introduced as a computer-based intelligent technique to deal with the problem of information and product overload. They can be utilized to efficiently provide personalized services in most e-business domains, benefiting both the customer and the merchant.

Starting in 1992, Tapestry [4] introduced Collaborative Filtering (CF), the most widely used recommender systems' algorithm. The initial CF algorithm was automated by GroupLens [11] and Ringo [16]. Content-based filtering, which draws ideas from the field of Information Filtering/Information Retrieval [12], was utilized as an alternative to classic CF. Most recently and based on the classic CF algorithm, Karypis [8] proposed Item-based Filtering. Many variations of these central ideas were suggested: Good [5] introduced intelligent filterbots. Balabanovic [1] and Melville [9] proposed methods that combine content-based with collaborative filtering. Sarwar [14] applied dimensionality reduction techniques. Breese [3] proposed Model-based algorithms like Cluster Models and Bayesian Network Models. Pennock [10] suggested CF by Personality Diagnosis as a hybrid Memory- and Model-based approach. Billsus [2] converted the recommendation process to a Machine Learning problem.

Our experiments will focus on two basic Memory-based filtering algorithms: classic CF and Item-based CF. Each of them views the recommendation task from a different perspective and it will be interesting to see how these perspectives are reflected in the results of our experiments.

The rest of the paper is organized as follows. The next section describes the utilized data set. Section 3 is split in four subsections. The first two subsections describe the experiments in classic CF and Item-based CF while discussing their results. The last subsection includes an overall method comparison. Finally, section 4 concludes the paper and provides directions for future research.

2 The Data Set

In order to execute our experiments we used the original GroupLens data set. The data set consists of 10.000 ratings, assigned by 943 users on 1682 movies. All ratings follow the 1 (bad) - 5 (excellent) numerical scale. Users and movies are accompanied by important information. Basic demographic data such as age, gender, occupation and zip code are included for all users. The following information exist for each movie in the set: Movie title, release date and video release date, IMDb URL, and finally, genre data, specifying if the movie is a comedy, a thriller, a documentary etc. The IMDb URL includes the web link of the corresponding movie to the biggest cinema related database on the internet (www.imdb.com). Finally, we have to note that the initial data set was used as the basis to generate five distinct splits into training and test data. For each split, 80% of the original set was included in the training and 20% of it was included in the test data. The test sets in all cases were disjoint. Such a separation was intended for the execution of *5 fold cross validation*, where all the experiments are run five times, once with each training and test set, and then the results are averaged. In our experiments those sets will be referred to as file=1, file=2, ..., file=5.

3 The Methods

3.1 Collaborative Filtering

In this section we will evaluate the utility of the classic Collaborative Filtering method. By this term we refer to the algorithm that bases its predictions on neighbours of relevant users. We will first provide a brief description of the various experiments we executed and then we will proceed and present the results of these experiments, trying to explain them when necessary.

Description of the Experiments The inspiration for classic Collaborative Filtering methods comes from the fact that people who agreed in their subjective evaluation of past items are likely to agree again in the future [11].

Construct a suitable Representation for the input data.
The classic Collaborative Filtering algorithm is based on the use of an *mxn user-item matrix, R.* For the purposes of our experiments we constructed two distinct but very similar arrays. The first array held the training data, i.e. the user ratings which were utilized for the creation of the neighborhoods of users and the prediction generation. The second array held the testing data, i.e. the actual ratings provided by the user against which the *predicted* ratings would be compared, in order to evaluate the predicting capabilies of the tested algorithm.

Compute the "similarity" between users in the user-item matrix, R.
In the first step of Neighborhood Formation, the similarity between all existing users should be calculated. That was achieved by repeatedly selecting a couple of users and applying the preferred proximity metric on them. Based on the results of past experiments in [3], the similarity metric we selected for our experiments was that of

Pearson Correlation Similarity. This whole procedure led to a new array including the similarity values between all users.

Proceed with the Neighborhood Generation for the active user.
The next stage of Neighborhood Formation involves the actual Neighborhood Generation for the active user, who is the person we would like to generate predictions for. Results from [13] have shown that the Center-based Neighborhood Formation algorithm should be the method of choice in the succeeding experiments.

Two techniques have been used to determine how many users to include in the active user's neighborhood: *Correlation Thresholding* and *Best n neighbors with Common Item Threshold*. The first technique sets an absolute correlation threshold. The value of the correlation between the active user and a second random user is expressed by the application of the Pearson Correlation Similarity metric. We select all neighbors whose absolute correlation to the active user is higher than the value of the given threshold and include them in his neighborhood. In our experiments we applied Correlation Thresholding for a series of different correlation thresholds.

The second technique does not simply pick the best n correlates, as expressed by the highest Pearson Correlation Similarity values, but it also requires that those users selected and the active user have rated in common a number of items that is higher than a specified threshold. By this common item threshold we make sure that a possibly high correlation between the active user and a second random user is based on an adequate number of common rated or purchased items. In our experiments we applied Best n neighbors with Common Threshold for (a) different values of neighborhood size n and (b) for various values of common item threshold.

Conclude with Prediction Generation based on the active user's neighborhood.
We decided to base our predictions on the approach taken by GroupLens [11], which is the one most widely used in Recommender Systems today. Specifically, it computes the average deviation of a neighbor's rating from the same neighbor's mean rating. The mean rating of a user is calculated based on all items rated by him. This average deviation can then be converted to fit the active user's rating distribution by adding it to the active user's mean rating.

Now, we can proceed and compare the ratings generated by the filtering algorithm with the actual ratings provided by the active user. The following section includes results from our classic Colaborative Filtering experiments and also discusses the possible causes of those results.

Description of the Results In the following paragraphs we will present and discuss classic Collaborative Filtering experiments' results:

Best n Neighbors with Common Item Threshold: Comparing different Neighborhood sizes
When applying *Best n Neighbors with Common Item Threshold*, the size of the neighborhood, as expressed by the value of n, has a crucial influence on the quality of the recommendation. In order to determine this impact we performed the following experiment where we assigned different values to the neighborhood size, starting from 10

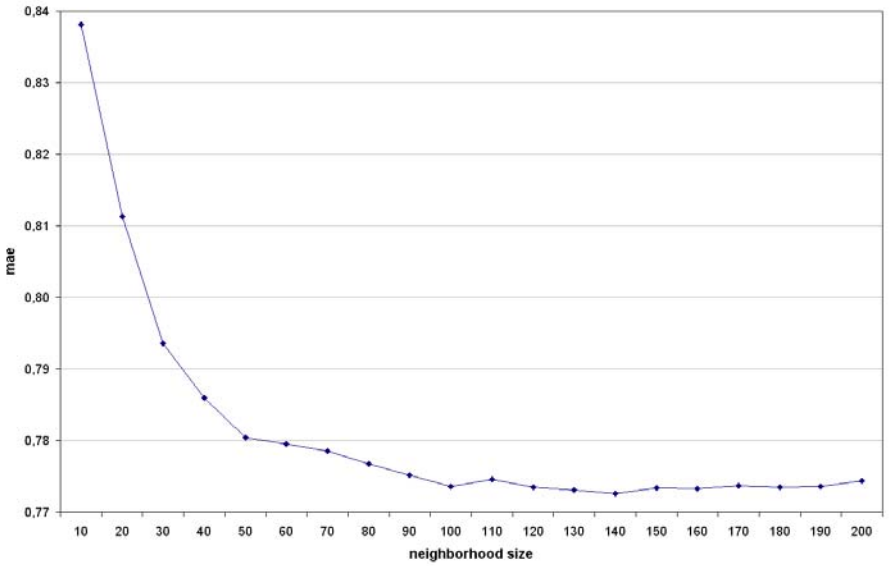

Fig. 1. Classic Collaborative Filtering: Comparing neighborhood sizes

users and finally reaching 120 users. During the experiment the Common Item Threshold was fixed to a value of 25, meaning that a user should have rated at least 25 movies in common with the active user in order to be included.

The mean absolute errors (mae) from this experiment averaged over all five data set splits are displayed in Figure 1.

Figure 1 verifies that the neighborhood size affects the accuracy of the recommendation. We can easily note that the quality of the results increases as the number of neighbors is increased. Nevertheless, there is a specific point above which the quality improvements are trivial or even non-existent. Based on those results we can assume that the optimal neighborhood includes 100 users.

Best n Neighbors with Common Item Threshold: Comparing different Common Item Thresholds

Once again we applied *Best n Neighbors with Common Item Threshold* in order to select which users to include in the active user's neighborhood. The difference was that this time we varied the Common Item Threshold while keeping the number of the neighbors fixed. We ran this experiment for a number of different neighborhood sizes so that we could see the relation between the common item threshold and the neighborhood size. Common item threshold values which were tested: $c\text{-}i\text{-}t=\{1, 10, 20, 30, 40\}$. These common item thresholds were utilized for the following neighborhood sizes: $n\text{-}s=\{10, 20, 30, 40, 50\}$. All these experiments were executed for a single data split (file=4) and the mean absolute errors can be found in Table 1.

	ave. mae	n-s=10	n-s=20	n-s=30	n-s=40	n-s=50
c-i-t=1	0,8885283735	1,03732555	0.88218964	0,84100155	0,86097659	0,86882881
c-i-t=10	0,7942188115	0,84677930	0,80564615	0,78630393	0,77571842	0,75664626
c-i-t=20	0,7763496282	0,83011702	0,77739673	0,76427949	0,75479415	0,75516074
c-i-t=30	0,7843445373	0,83455159	0,78052642	0,77641101	0,76556608	0,76466758
c-i-t=40	0,7853143170	0,81315102	0,78691052	0,77294615	0,78127219	0,77229170

Table 1. Comparing mae for various common item thresholds

As one can see, the error has its highest value for *c-i-t=1*. This result was not unusual since in that case the proximity between two users is based upon a *single* commonly rated item, which makes any claim of similarity particularly weak. The error then gradually decreases and gets its lowest value for *c-i-t=20*. After that point, for values of *c-i-t* > 20, the error starts increasing again. We can explain this behaviour simply by saying that when the common item threshold gets higher values, it is required that many commonly rated items exist between two users in order for their similarity to be calculated. This results in a limited number of users who are allowed to be included in the active user's neighborhood. The small user neighborhoods which are then produced lead to recommendations of low quality.

Table 2 includes the average Coverage values achieved, over all different neighborhood sizes tested, for the same experiment. This table makes clear that the behavior

	ave. coverage
c-i-t=1	47,1538832610%
c-i-t=10	80,8248914616%
c-i-t=20	84,6840328027%
c-i-t=30	82,0308731307%
c-i-t=40	79,9083453932%

Table 2. Comparing coverage from various common item thresholds

of the average coverage values is very similar to the behavior of the error values. The worst coverage is once again met for *c-i-t=1*. The highest coverage value is observed for *c-i-t=20*. For values of *c-i-t* > 20 the coverage starts decreasing again. The behavior of both mean absolute error and coverage lead us to the conclusion that a common item threshold value of 20 should be preferred.

Correlation Thresholding: Comparing different Correlation Thresholds

In our third experiment we applied Correlation Thresholding in order to generate the active user neighborhood, which means that all neighbors with absolute correlations to the active user higher than the given threshold were included in his neighborhood. We tested the following correlation threshold values: *corr-t*=$\{$*0.1, 0.2, 0.3, 0.4, 0.5*$\}$. The

results from this experiment were averaged for all five data set splits and can be found in Table 3. All numbers shown correspond to mean absolute errors.

	ave. mae
corr-t=0.1	0,7718232669
corr-t=0.2	0,7718509556
corr-t=0.3	0,7725067723
corr-t=0.4	0,7872393620
corr-t=0.5	0,8149313812

Table 3. Comparing mae from various correlation thresholds

Table 4 includes the average Coverage values achieved, over all five data splits tested, for the same experiment.

	ave. coverage
corr-t=0.1	99,7397417006%
corr-t=0.2	99,5752868495%
corr-t=0.3	98,8603343632%
corr-t=0.4	96,5569821610%
corr-t=0.5	88,2237317341%

Table 4. Comparing coverage from various correlation thresholds

When we set high correlation thresholds, *corr-t={0.4, 0.5}*, we actually require from our neighborhoods to include users that tend to be really close to the active user. We assumed that this fact would lead us to predictions of improved quality when compared to those with lower correlation thresholds, *corr-t={0.1, 0.2, 0.3}*. Still, in a rather surprising result the quality of the predictions was actually better for the lower correlation threshold value which we utilized, *corr-t=0.1*. The related literature [6] reports similar unexpected results. One probable explanation: Those neighbors selected for high correlation thresholds are normally very similar to the target user. Still, just because they agree with the target user does not necessarily imply that they should agree with each other. Furthermore, their possible contradicting with each other would be weighted heavily because of their high correlations values. As a result, the total recommendation quality would be brought down.

On the other hand, the behavior of the coverage results was expected. High correlation values force the user neighborhood to include only good correlates, but it is possible that for specific users such correlates simply do not exist. This will lead to small neighborhoods that cannot provide prediction coverage for many items, a fact that is clearly reflected in the way the coverage results drop as the correlation threshold is increased.

3.2 Item-based Collaborative Filtering

In this section we will discuss the utility of Item-based Collaborative Filtering, an algorithm proposed recently in the Recommender Systems literature [15] [8].

Description of the Experiments Similarly to classic Collaborative Filtering, Item-based Filtering is based on the creation of neighborhoods. Unlike classic Collaborative Filtering, those neighbors consist of similar items rather than similar users.

Construct a suitable Representation for the input data.
The Item-based Collaborative Filtering algorithm, like classic Collaborative Filtering, is based on the use of an *mxn user-item matrix, R*. So, again we constructed two distinct but very similar arrays. The first array contained the training data, i.e. the user ratings which were utilized for the creation of the neighborhoods of items and the prediction generation. The second array included the testing data, i.e. the actual ratings provided by the user. *Predicted* ratings would then be compared against them, in order to evaluate the predicting capabilities of the tested algorithm.

Compute the "similarity" of items in the user-item matrix, R
To achieve item similarity calculation we repeatedly selected couples of *items* and applied the desired proximity metrics on them. In our case we wanted to compare the two main similarity computation measures described in [15]: *Pearson Correlation Similarity* is a measure very similar to the one utilized in classic Collaborative Filtering for user proximity calculation. *Adjusted Cosine Similarity* is a new measure applied exclusively in Item-based Collaborative Filtering.

Proceed with the Neighborhood Generation for the active item.
Once the correlations between items are specified and stored in the corresponding array, the following step is to select only those items that will be included in active item's neighborhood. The technique we applied to accomplish that effect is called *Best n neighbors with Common User Threshold*. Based on this method we pick the best n correlates, expressed by the highest proximity measure values, but at the same time we require that those items selected and the active item have been rated by a number of common users that is higher than a specified threshold. In our experiments we utilized Best n neighbors with Common Threshold for (a) various values of neighborhood size n and (b) for different values of common user threshold.

Conclude the Recommendation Process by Prediction Generation based on the active item's neighborhood.
We based our final predictions on the conventional approach proposed in [15], where a weighted sum of ratings given by a user on all items similar to the active item is calculated. Still, our initial experiments revealed a possible problem with this formula: Enough negative correlates included in the item neighborhood may lead to negative weighted sums and correspondingly to negative ratings. Negative predictions are unacceptable based on the utilized rating scale, but at the same time they considerably degrade the system's performance, when they are compared with the actual, and always positive, ratings. To solve this problem we applied a very straightforward solu-

tion: Whenever a prediction, based on the weighted sum formula, was negative, we forced it to a value of 1, i.e. the minimum legal rating value. We have included a relevant experiment where the impact of this simple correction was evaluated for both Correlation-based and Adjusted Cosine similarity measures.

Once the filtering algorithm produces its predictions, we can compare them with the actual ratings provided by the active user.

Description of the Results Results and conclusions extracted by experiments executed on Item-based filtering are included in the following sections.

Methods for Evaluating Similarity: Correlation-based vs. Adjusted Cosine

The purpose of our first experiment in Item-based Collaborative Filtering was to contrast two distinct item similarity measures: Pearson Correlation (cor-based) and Adjusted Cosine (adj cos) Similarity. We collected results from a single data split (file=4) for varying neighborhood sizes, *n-s={10, 20, 30, 40, 50, 60, 80, 100}*. Table 5 displays the mean absolute errors when utilizing the tested similarity metrics for those parameters:

	Cor-based	**Adj Cos**
n-s=10	1,3331658344	0,8627412979
n-s=20	1,2704950185	0,8156449079
n-s=30	1,2562016265	0,8060186623
n-s=40	1,1822869516	0,8043008572
n-s=50	1,1631403743	0,7965577361
n-s=60	1,1432580376	0,7969093095
n-s=80	1,0798463900	0,7960940712
n-s=100	1,0532607636	0,7950274398
average	1,1852068746	0,8091617852

Table 5. Correlation-based vs. Adjusted Cosine: Comparing mae

With a minimum error value equal to 0,7950274398 and an average error value equal to 0,8091617852, Adjusted Cosine Similarity clearly outperforms Pearson Correlation (minimum error=1,0532607636, average error=1,1852068746) when accuracy is concerned.

Table 6 includes the Coverage values observed for the same experiment. Once again we can realize that Adjusted Cosine Similarity should be our proximity measure of choice. Its coverage peaks at values above 95% while at the same time Pearson Correlation's coverage, for the same neighborhood sizes, reaches at most 79%. As a result, the following experiments in Item-based Collaborative Filtering will be executed by utilizing Adjusted Similarity for all our correlation calculations.

	Cor-based	Adj Cos
n-s=10	24,3125904486%	71,9247467438%
n-s=20	37,0477568741%	83,7916063676%
n-s=30	47,4674384949%	89,2908827786%
n-s=40	54,2691751085%	92,1852387844%
n-s=50	59,4790159190%	93,3429811867%
n-s=60	64,6888567294%	94,0665701881%
n-s=80	73,5166425470%	95,0795947902%
n-s=100	79,0159189580%	95,5137481910%
average	54,9746743849%	89,3994211288%

Table 6. Correlation-based vs. Adjusted Cosine: Comparing coverage

Best n Neighbors with Common User Threshold: Comparing different Neighborhood sizes

Through this experiment we wanted to evaluate how variations in the size of the active item's neighborhood affected the quality of the recommendation, when applying *Best n Neighbors with Common User Threshold*. This impact was determined by assigning different values to the neighborhood size, *n-s={10, 20, 30, 40, 50, 60, 70, 80, 90, 100}*, while keeping the value of common user threshold fixed to 10, meaning that at least 10 users should have rated both the active item and a second random item, in order for the latter to be considered.

The resulting mean absolute errors from this experiment, averaged over all five data splits, can be found in Figure 2.

From Figure 2 we can see that the error keeps dropping while the size of the neighborhood is increased. It then reaches its lowest value for neighborhood sizes in the range of 70 to 80 items. After this point, the error values stay fixed and soon start increasing, making those neighborhood sizes undesirable. A possible explanation of this behaviour: Certain items which are included only in the bigger neighborhoods, being absent before, may not be that relevant to the active item, and as a result lead to this performance degrading.

Figure 3 includes the average Coverage values achieved, over all five data splits tested, for the same experiment.

As expected, higher neighborhood sizes lead to higher coverage - even if that, as the previous figure verified, does not necessarily translate to lower errors. This trend is apparent for small neighborhood sizes, *n-s={10, 20, 30, 40}*, when the improvements in coverage are big, but also continues for higher neighborhood sizes, *n-s={70, 80, 90, 100}*, when the increases in coverage are trivial.

For this experiment, a neighborhood of 100 items achieved the highest coverage at 95,4208155814%. Still, the coverage value for *n-s=70*, which was the neighborhood size with the lowest error, was very close at 94,3816278413%. As a result, we conclude that the optimal value for the neighborhood size is around 70 for the utilized data sets.

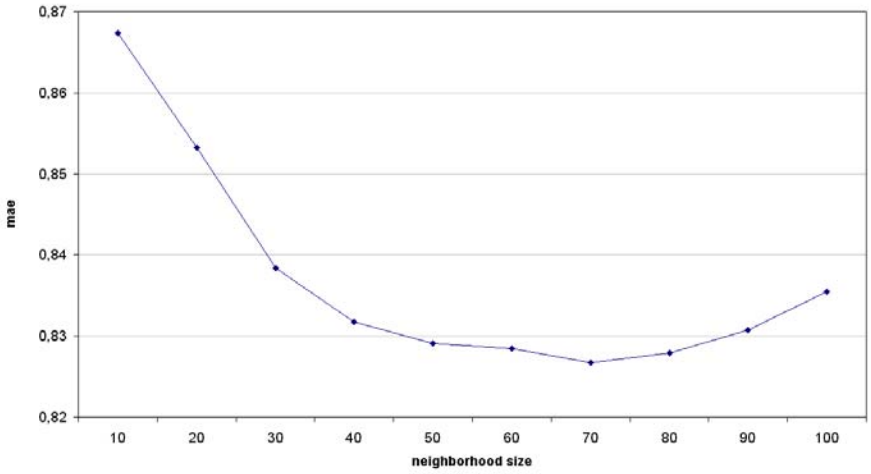

Fig. 2. Item-based Filtering: Comparing mae from various neighborhood sizes

Best n Neighbors with Common User Threshold: Comparing different Common User Thresholds

Once again, we applied *Best n Neighbors with Common User Threshold* but this time we intended to reach conclusions regarding the role of Common User Threshold. To achieve that effect in our experiment we kept the neighborhood size fixed and varied the values of Common User Threshold. Common User Threshold values which we tested are: *c-u-t={1, 10, 20, 30, 40}*. In order to locate any existing relations between Common User Threshold and Neighborhood size, we ran the experiment for item neighborhoods of the following sizes: *n-s={10, 10, 20, 30, 40, 50}*. A single data split was used (file=4) and the absolute error values can be found in Table 7.

	ave. mae	**n-s=10**	**n-s=20**	**n-s=30**	**n-s=40**	**n-s=50**
c-u-t=1	1,02256103	1,13379628	1,05670956	0,98478628	0,97130966	0,9662034
c-u-t=10	0,81705269	0,86274130	0,81564491	0,80601866	0,80430086	0,7965577
c-u-t=20	0,81520353	0,80318379	0,82485771	0,80865204	0,81433707	0,8249870
c-u-t=30	0,84262647	0,81785558	0,80124272	0,83389472	0,86737594	0,8927634
c-u-t=40	0,86777971	0,80640416	0,81471545	0,86637725	0,90085551	0,9505462

Table 7. Comparing mae for various common user thresholds

Table 8 includes the average Coverage values over all different neighborhood sizes we tested, for the same experiment.

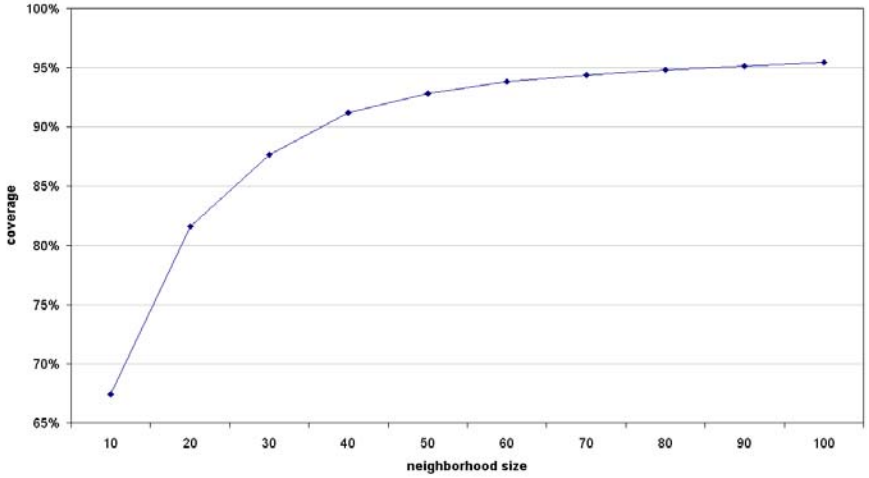

Fig. 3. Comparing coverage from various neighborhood sizes

	ave. coverage
c-u-t=1	55,94376680%
c-u-t=10	88,52594583%
c-u-t=20	89,04760815%
c-u-t=30	84,24643374%
c-u-t=40	76,49369444%

Table 8. Comparing coverage for various common user thresholds

A careful observation of both tables leads us to the conclusion that average error and coverage values show identical behavior: They have their worst value - a highest error of 1,02256103 and a lowest coverage of 55,9437668% - when *c-u-t=1*. Such a low Common User Threshold value cannot lead to truly similar items and generates inaccurate results. Their best value is achieved when *c-u-t=20*. For values of $c\text{-}u\text{-}t > 20$ we see that coverage starts decreasing while error values increase. We attribute that to the fact that high Common User Threshold values lead to small item neighbhorhoods, which result to recommendations of lower quality.

An interesting finding based on Table 7 has to do with the correlation between Common User Threshold and Neighborhood Size. One can see that for low neighborhood sizes, *n-s={10, 20}*, the error values have a smooth decrease as Common User Threshold gets higher. They reach their lowest value approximately for *c-u-t=20* and then very slowly they start increasing. For bigger neighborhood sizes, *n-s={30, 40, 50}*, we see that the error values show a very different behavior: They almost immediately reach their lowest value, sometimes even for $c\text{-}u\text{-}t < 20$ and then they display an

abrupt rise. It is noteworthy that for *n-s=50*, the highest neighborhood size we experimented with, the error reached its minimum value (0,7965577) unexpectedly fast, for *c-u-t=10*, and then instantly got back to high, unacceptable values, such as 0,9505462 for *c-u-t=40*.

We should point out that the same experiment (*Comparing different Item Thresholds*), when executed for classic Collaborative Filtering, displayed a very similar behavior for both mean absolute error and coverage. Concluding, for the specific data set we suggest a Common User Threshold equal to 20 when applying Best n Neighbors with Common User Threshold in order to generate item neighborhoods.

Comparing Error with Prediction Correction Included and Not Included

While describing our experiments, we talked about the necessity to add a correction to the prediction calculation formula in order to avoid negative predictions. The aim of this experiment is to evaluate the impact of the simple correction we proposed when either Correlation-based or Adjusted Cosine similarity measures are utilized. We ran the experiment for various neighborhood size values, *n-s={10, 20, 40, 60, 80, 100}*. Table 9 collects the error values of Correlation-based similarity **without** the correction (termed *cor wo/ correct.*) and **with** the correction (termed *cor w/ correct.*). It also includes the error values of Adjusted Cosine similarity **without** the correction (termed *adj wo/ correct.*) and **with** the correction (termed *adj w/ correct.*).

	cor wo/ correct.	**cor w/ correct.**	**adj wo/ correct.**	**adj w/ correct.**
n-s=10	1,3331658344	1,3331658344	0,9565081230	0,862741298
n-s=20	1,2704950185	1,2704950185	0,9055895944	0,815644908
n-s=40	1,1882942835	1,1822869516	0,9071966216	0,804300857
n-s=60	1,1485521534	1,1432580376	0,9176989950	0,796909309
n-s=80	1,0845047950	1,0798463900	0,9213020768	0,796094071
n-s=100	1,0575949573	1,0532607636	0,9300166973	0,79502744
average	1,1804345070	1,1770521660	0,9230520180	0,8117863139

Table 9. Comparing similarity measures with and without correction

Table 9 shows that for the case of Pearson Correlation Similarity, the application of the prediction correction leads to a rather insignificant improvement: The average error **without** the correction is 1,1804345070, while the average error **with** the correction is a bit lower at 1,1770521660.

On the other hand, it is certainly more interesting to see how our proximity measure of choice, Adjusted Cosine Similarity, reacts to the application of the correction. It is obvious, for all the neighborhood sizes we tested, that when the prediction correction is applied, the accuracy improvement is considerable: The average error **without** the correction is 0,9230520180, while the average error **with** the correction gets down to 0,8117863139. This comparison verifies the claim that the weighted sum formula described in [15] for prediction generation is flawed and requires further refining or

clarification. The simple correction we proposed did actually improve prediction accuracy, and as a result it can be adopted. Nevertheless it would be wise to refine the existing formula in order to avoid extreme cases or even to implement a more intelligent improvement that would solve the observed problem.

3.3 Overall Recommender Algorithms' Comparison

For each of the two approaches we implemented and tested, a big number of results was generated, all of which were included and analyzed in previous sections. In this section, we simply set the parameters accordingly our only purpose being to achieve the best performing classic Collaborative Filtering (*best CF*) and the best Item-based Collaborative Filtering (*best item*). Those results were then contrasted to a simpler but faster "smart" non-personalized algorithm, (*non-person*). The non-personalized approach we implemented is "smart" because it does not compute the average rating of the item being predicted over all users. Instead, it computes the deviation-from-mean average over all users [7].

	average mae
best CF	0,77342001
best item	0,82674183
non-person	0,81562274

Table 10. Overal error comparison

Table 10 summarizes the mean absolute errors obtained when applying all those filtering algorithms for optimal parameter values. The filtering method with the best results - lower mean absolute error on average equal to 0,77342001 - is the classic Collaborative Filtering. In a rather surprising result, the smart non-personalized approach we implemented came second (error: 0,81562274), with Item-based Collaborative Filtering a close third (error: 0,82674183). We should make a special reference to the case of Item-based Filtering. Related papers stated that Item-based Collaborative Filtering can provide recommendations with quality up to 27% better than the traditional user-neighborhood based recommender systems [8]. We could not repeat such good results in any of our Item-based experiments, regardless of all different parameter settings we utilized. Our experiments showed that the recommender algorithm of choice should still be classic Collaborative Filtering.

4 Conclusions

In this work we chose two basic filtering algorithms used extensively in commercial and research Recommender Systems. We executed a series of experiments in order to evaluate them. The selection of the algorithms was made so that they would represent contrasting ways of tackling the recommendation problem. Specifically, the classic

Collaborative Filtering algorithm focuses on similar users as judged by their ratings, collecting them in user neighborhoods, whereas Item-based Filtering focuses on similar items as judged by user ratings, collecting them in item neighborhoods.

After breaking those filtering algorithms into their successive steps, we tried different parameter settings for each one of them, in order to see how the algorithms respond, finally reaching optimal settings. This process sometimes led us to unexpected results which we had to investigate, as in the case of the prediction generation formula of Item-based Filtering. Then we proceeded to contrast the methods' performance against each other.

Classic Collaborative Filtering was proven to be the best performing algorithm. The smart non-personalized approach performed surprising well. Item-based Filtering displayed disappointing behavior. Despite opposite claims, did not only perform worse than classic Collaborative Filtering, but also worse than the non-personalized approach.

Further work is required to understand why Item-based performed this way. Also, we intend to further investigate possible uses of Machine Learning algorithms and dimensionality reduction techniques in order to assist the recommendation process.

References

1. M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40, 1997.
2. Daniel Billsus and Michael J. Pazzani. Learning collaborative filters. In *Proceedings 15th Conference on Machine Learning*, Madison, WI, 1998.
3. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings 14th Conference on Uncertainty in Artificial Intelligence (UAI)*, Madison, WI, 1998.
4. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
5. Nathaniel Good, J. Ben Schafer, Joseph A. Konstan, Al Borchers, Bardul M. Sarwar, Jon Herlocker, and John T. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings Conference of the American Association of Artificial Intelligence (AAAI)*, pages 439–446.
6. Jon Herlocker, Joseph A. Konstan, Al Borchers, and John T. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings Conference on Research and Development in Information Retrieval*, 1999.
7. Jonathan Lee Herlocker. *Understanding and Improving Automated Collaborative Filtering Systems.* PhD thesis, University of Minnesota, 2000.
8. George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings CIKM Conference*, 2001.
9. Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering. In *Proceedings ACM SIGIR Workshop on Recommender Systems*, New Orleans, LA, 2001.
10. David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybric memory- and model-based approach. In *Proceedings 16th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 473–480, San Francisco, CA, 2000.

11. Paul Resnick, Neophytos Iacovou, Mitesh Sushak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings ACM Conference on Computer Supported Cooperative Work*, pages 175–186, New York, NY, 1994.
12. Gerard Salton and Christopher Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
13. Bardul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Analysis of recommendation algorithms for e-commerce. In *Electronic Commerce*, 2000.
14. Bardul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender systems - a case study. In *Proceedings ACM WebKDD Web Mining for E-Commerce Workshop*, 2000.
15. Bardul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings 10th World Wide Web Conference (WWW)*, Hong Kong, 2001.
16. Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings Computer Human Interaction Conference*, pages 210–217, 1995.