Designing Specific Weighted Similarity Measures to Improve Collaborative Filtering Systems

Laurent Candillier, Frank Meyer, Françoise Fessant
France Telecom R&D Lannion, France
{name.surname}@orange-ftgroup.com

Abstract. The aim of *collaborative filtering* is to help *users* to find *items* that they should appreciate from huge catalogues. In that field, we can distinguish *user-based* from *item-based* approaches. The former is based on the notion of user neighbourhoods while the latter uses item neighbourhoods.
The definition of *similarity* between users and items is a key problem in both approaches. While traditional similarity measures can be used, we will see in this paper that bespoke ones, that are tailored to type of data that is typically available (i.e. very sparse), tend to lead to better results.
Extensive experiments are conducted on two publicly available datasets, called *MovieLens* and *Netflix*. Many similarity measures are compared. And we will show that using weighted similarity measures significantly improves the results of both user- and item-based approaches.

1 Introduction

There has been a growth in interest in *recommender systems* [1] in the last two decades, since the appearance of the first papers on this subject in the mid-1990s [2]. The aim of such systems is to help *users* to find *items* that they should appreciate from huge catalogues.

Items can be of any type, such as movies, music, books, web pages, online news, jokes and restaurants. In this paper, we focus on movie items. The goal, therefore, of recommender systems, in this context, is to help users to find movies of interest, based on some information about their historical preferences.

Three types of recommender systems are commonly proposed:

1. *collaborative filtering*;
2. *content-based filtering*;
3. and *hybrid filtering*.

In the first case, the input to the system is a set of user ratings on items. Users can be compared based upon their shared appreciation of items, creating the notion of user neighbourhoods. Similarly, items can be compared based upon the shared appreciation of users, rendering the notion of item neighbourhoods. The item rating for a given user can then be predicted based upon the ratings given in her user neighbourhood and the item neighbourhood.

In content-based filtering, however, item content descriptions are used to construct *user thematic profiles*, that contain information about user preferences,

such as, in the context of movies, "*like comedy and dislike war*". The user's predicted appreciation of a given item is then based on the proximity between the item description and the user profile.

Finally, in the case of hybrid filtering, both types of information, collaborative and content-based, are exploited.

Collaborative filtering techniques are more often implemented than the other two and often result in better predictive performance. The main reason is that they do not require these difficult to come by, well-structured item descriptions. Instead, they are based on users' preferences for items, that can carry a more general meaning than is contained in an item description. Indeed, viewers generally select a movie to watch based upon more elements than only its genre, actors and director.

So this paper focuses on collaborative filtering, that can be divided into two sets of approaches:

1. *user-based* approaches [2] associate a set of nearest neighbours with each user, and then predict the user's rating for unscored items using the ratings given by the neighbours on that item;
2. and *item-based* approaches [3] associate an item with a set of nearest neighbours, and then predict the user's rating for an item using the ratings given by the user on the nearest neighbours of the target item.

The definition of similarity between users and items is a key problem in each approach. Since the data that are typically available in collaborative filtering are very sparse, traditional similarity measures need to be adapted. Generally, when comparing two users or two items, only the set of attributes in common are considered. However, by doing so, many users and items may be compared based only on very few attributes, which can lead to a lack of meaning.

In this paper, we propose to adapt traditional similarity measures to sparse data. Our approach consists in using a new weighting scheme of existing similarity measures, so that users and items that share many attributes are prefered to others that only share a few attributes. We will then show the interest of such new measures, not only in improving the predictive performance of collaborative filtering systems, but also in improving their scalability.

This brings us naturally to the issue of evaluating the performance of a given recommender system [4]. This is typically done by using *cross-validation*. Then the most widely used measures for comparison are:

1. *Mean Absolute Error* (MAE);
2. *Root Mean Squared Error* (RMSE);
3. and *Precision* and *Recall*.

The first two measures evaluate the capability of a method to predict if a user will like or dislike an item, whereas the third set of measures evaluates its capability of providing an ordered list of items that a user should like. These measures thus carry different meanings [5]. In the first two cases, the method needs to be able to predict dislike, but there is no need to order items. In the

last case, however, the method only focuses on items that users will like and the order in which these items are ranked is important.

Beyond the importance of the predictive performance of recommender systems, other elements may be taken into consideration in their evaluation. The scalability of the proposed system is for example an important characteristic that needs to be taken into account.

To conduct experiments, we will use two real rating datasets that are publicly available, called *MovieLens* and *Netflix* [6]. The first one contains 1,000,209 ratings collected from 6,040 users on 3,706 movies. The second dataset contains 100,480,507 ratings for 480,189 users and 17,770 movies. Both are completely independent.

To summarise, this paper, on the study of similarity measures for collaborative filtering, will be structured as follows: an overview of the principal approaches for collaborative filtering is presented in section 2, alongside a study of similarity measures for comparing users and items; the results of extensive experiments are reported in section 3 in order to compare various similarity measures for collaborative filtering approaches, using cross-validation on the MovieLens and Netflix datasets; finally, section 4 concludes the paper and topics for future research are suggested.

2 Collaborative Filtering

Let U be a set of N users, I a set of M items, and R a set of ratings r_{ui} of users $u \in U$ on items $i \in I$. $S_u \subseteq I$ stands for the set of items that user u has rated.

Let us set to min_r the minimum rating and Max_r the maximum rating. In the MovieLens and Netflix datasets for example, ratings are integers ranging from 1 (meaning dislike) to 5 (meaning high like).

The goal of collaborative filtering approaches is then to be able to predict the rating p_{ai} of a given user a on a given item i. User a is supposed to be *active*, meaning that she has already rated some items, so $S_a \neq \emptyset$. But the item to be predicted shall not be known by the user, so $i \notin S_a$.

2.1 User-based approaches

For user-based approaches [2, 7], the prediction of a user rating on an item is based on the ratings, on that item, of the nearest neighbours. So a similarity measure between users needs to be defined. Then a set of nearest neighbours is selected. And finally, a method for combining the ratings of those neighbours on the target item needs to be chosen.

The way the similarity between users is computed will be discussed in subsection 2.3. Let $sim(a, u)$ be that similarity measure between users a and u. The number of neighbours considered is often set by a system parameter that we denote by K. So the set of neighbours of a given user a, denoted by T_a, is made of the K users that maximise their similarity to user a.

A possible way to predict the rating of user a on item i is then to use the weighted sum of the ratings of the nearest neighbours $u \in T_a$ that have already rated item i:

$$p_{ai} = \frac{\sum_{\{u \in T_a | i \in S_u\}} sim(a,u) \times r_{ui}}{\sum_{\{u \in T_a | i \in S_u\}} |sim(a,u)|} \tag{1}$$

In order to take into account the difference in use of the rating scale by different users, predictions based on deviations from the mean ratings have been proposed. In that case, p_{ai} is computed using the sum of the user mean rating and the weighted sum of deviations from their mean rating of the neighbours that have rated item i:

$$p_{ai} = \overline{r_a} + \frac{\sum_{\{u \in T_a | i \in S_u\}} sim(a,u) \times (r_{ui} - \overline{r_u})}{\sum_{\{u \in T_a | i \in S_u\}} |sim(a,u)|} \tag{2}$$

$\overline{r_u}$ represents the mean rating of user u:

$$\overline{r_u} = \frac{\sum_{\{i \in S_u\}} r_{ui}}{|S_u|} \tag{3}$$

Indeed, let us suppose a user rates 4 a movie she likes and 1 a movie she dislikes while another user rates 5 a movie she likes and 2 a movie she dislikes. Then using deviations from their mean rating will reduce the effect of such a difference in use of the rating scale.

The time complexity of user-based approaches is $O(N^2 \times M \times K)$ for the neighbourhood model construction, it is $O(K)$ for one rating prediction, and the space complexity is $O(N \times K)$.

2.2 Item-based approaches

Then item-based approaches have known a growing interest [3, 8–10]. Given a similarity measure between items, such approaches first define item neighbourhoods. Then the rating of a user on an item is predicted by using the ratings of the user on the neighbours of the target item.

The possible choices of the similarity measure $sim(i,j)$ defined between items i and j will be discussed in the next subsection. Then, as for user-based approaches, the item neighbourhood size K is a system parameter that needs to be defined. And given T_i the neighbourhood of item i, two ways for predicting new user ratings can be considered: weighted sum, and weighted sum of deviations from the mean item ratings:

$$p_{ai} = \frac{\sum_{\{j \in S_a \cap T_i\}} sim(i,j) \times r_{aj}}{\sum_{\{j \in S_a \cap T_i\}} |sim(i,j)|} \tag{4}$$

$$p_{ai} = \overline{r_i} + \frac{\sum_{\{j \in S_a \cap T_i\}} sim(i,j) \times (r_{aj} - \overline{r_j})}{\sum_{\{j \in S_a \cap T_i\}} |sim(i,j)|} \tag{5}$$

$\overline{r_i}$ represents here the mean rating on item i:

$$\overline{r_i} = \frac{\sum_{\{u \in U | i \in S_u\}} r_{ui}}{|\{u \in U | i \in S_u\}|} \tag{6}$$

The time complexity of item-based approaches is $O(M^2 \times N \times K)$ for the neighbourhood model construction, it is $O(K)$ for one rating prediction, and the space complexity is $O(M \times K)$.

2.3 Similarity measures

The similarity defined between users or items is crucial in collaborative filtering. The first one proposed in [2] is *pearson* correlation. It corresponds to the cosine of deviations from the mean:

$$pearson(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a}) \times (r_{ui} - \overline{r_u})}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a})^2 \sum_{\{i \in S_a \cap S_u\}} (r_{ui} - \overline{r_u})^2}} \tag{7}$$

Simple *cosine* can also be used. It has been tested in [3]:

$$cosine(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} r_{ai} \times r_{ui}}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} r_{ai}^2 \sum_{\{i \in S_a \cap S_u\}} r_{ui}^2}} \tag{8}$$

Other similarity measures based on cosine have been proposed. But they have been shown less relevant in [11]. We also propose here the simple case of *manhattan* similarity:

$$manhattan(a,u) = 1 - \frac{1}{Max_r - min_r} \times \frac{\sum_{\{i \in S_a \cap S_u\}} |r_{ai} - r_{ui}|}{|\{i \in S_a \cap S_u\}|} \tag{9}$$

For all these similarity measures, only the set of attributes in common between two vectors are considered. Thus two vectors may be completely similar even if they only share one appreciation on one attribute.

Let us illustrate the drawback of such measures on one example. One user is fan of science fiction while another user only watches comedys. They haven't rated any movie in common so their similarity is null. Now they both say they like "*Men In Black*", a comic science fiction movie. So these two users become completely similar according to the previously presented measures.

On the contrary, *jaccard* similarity doesn't suffer from this limitation since it measures the overlap that two vectors share with their attributes:

$$jaccard(a,u) = \frac{|\{S_a \cap S_u\}|}{|\{S_a \cup S_u\}|} \tag{10}$$

On the other hand, such a measure doesn't take into account the difference of ratings between the vectors. In this case, if two users watch the same movies

but have completely opposite opinions on them, then they are considered to be similar anyway according to jaccard similarity.

So we propose to combine jaccard with the other similarity measures, in order to benefit from their complementarity. We propose to simply compute the product between the jaccard similarity and the other ones. Jaccard would thus serve as a weighting scheme. So *wpearson* stands for *weighted pearson*, the new similarity measure made by the product of pearson and jaccard. Similarly, we denote by *wcosine* and *wmanhattan* the combination of jaccard with cosine and manhattan respectively.

Cosine-based similarity measures have their values comprised between -1 and 1 while the other similarity values are comprised between 0 and 1.

3 Experiments

3.1 Protocol

We conduct these experiments using the MovieLens and Netflix datasets. We divide them into 2 parts in order to perform cross-validation, training the chosen model using 90% of the datasets and testing it on the last 10%. In all experiments, the division of the datasets is always the same, so that all approaches are evaluated under exactly the same conditions. Since the dataset sizes are important, the variance of the results over different cuts of the datasets is low.

Given $T = \{(u, i, r)\}$ the set of (user, item, rating) triplets used for test, the Mean Absolute Error Rate (MAE) and Root Mean Squared Error (RMSE) are used to evaluate the performance of the algorithms:

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{ui} - r| \tag{11}$$

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i,r) \in T} (p_{ui} - r)^2} \tag{12}$$

The predicted ratings are rounded when the MAE is reported. First of all, this improves the results. Besides, rounding ratings is natural in practice, since real users generally prefer rating scales based on natural numbers than on real numbers.

We then use precision measures specifically designed for the current context. precision_5 stands for the proportion of maximum ratings in the test dataset ($Max_r = 5$) that are retrieved as the best predicted ratings. Similarly, precision_4 stands for the proportion of test ratings higher than $Max_r - 1 = 4$ that are considered as the best ratings using the given recommender system.

Finally, we also report the time spent to learn the models, and for predictions. The computer used for these experiments has 32Go RAM and 64 bits 3.40GHz 2-cores CPU.

3.2 Results

Considering only the principal collaborative filtering approaches already leads us to a lot of choices and parameters. When implementing a user- or item-based approach, one may choose:

- a similarity measure: pearson (equation 7), cosine (8), manhattan (9), jaccard (10), or the proposed combinations of jaccard with the others;
- a neighbourhood size K;
- and how to compute predictions: using a weighted sum of rating values (equations 1 and 4), or using a weighted sum of deviations from the mean (2 and 5).

Prediction scheme based on deviations from the mean has been shown to be more effective in [11]. We confirmed this result in our experiments. So in the following, only the results using this scheme are reported.

This paper is about the comparison of various similarity measures. So the following figures 1 to 3 show the Mean Absolute Error Rate obtained using the proposed measures, varying the neighbourhood size K from 10 to the maximum number of possible neighbours, for both user- and item-based approaches, and on both MovieLens and Netflix datasets.

Figure 1 first shows the error rates of item-based approaches depending on the similarity measure used and the neighbourhood size. The optimum is reached with the weighted pearson similarity and 100 neighbours. All similarity measures are improved when they are weighted with jaccard, at least when few neighbours are considered. All these weighted similarity measures reach their optimum when 100 neighbours are selected. On the contrary, non-weighted similarity measures need much more neighbours to reach their optimum. 700 neighbours shall be selected when using simple manhattan similarity, and 1500 when using simple cosine or simple pearson.

Figures 2 and 3 show that the same conclusions hold when using user-based approaches, as well as when the Netflix dataset is used instead of MovieLens. Weighted pearson similarity always leads to the best results. Using our proposed weighting scheme always improves the results. 300 neighbours shall be considered for an user-based approach on MovieLens, and 70 for an item-based approach on Netflix. On the contrary, 2000 to 4000 neighbours need to be selected to reach the minimum error rate of non-weigthed similarity measures.

The results have been presented for the MAE. They are completely similar for the other performance measures: RMSE and precisions. Beyond the improvement of predictive performance when our proposed weighting scheme is used, another important advantage is that fewer neighbours need to be selected, so that the algorithms also gain in scalability.

Tables 1 and 2 summarise the results of the best of each approach, including learning and prediction times, and precisions. The default approach that is reported is Bayes designed to minimise RMSE. More details on this method can be found in [11]. Both user- and item-based approaches reach optimal results when the weighted pearson similarity is used. On MovieLens, 300 neighbours are selected for the best user-based approach, and 100 for the best item-based

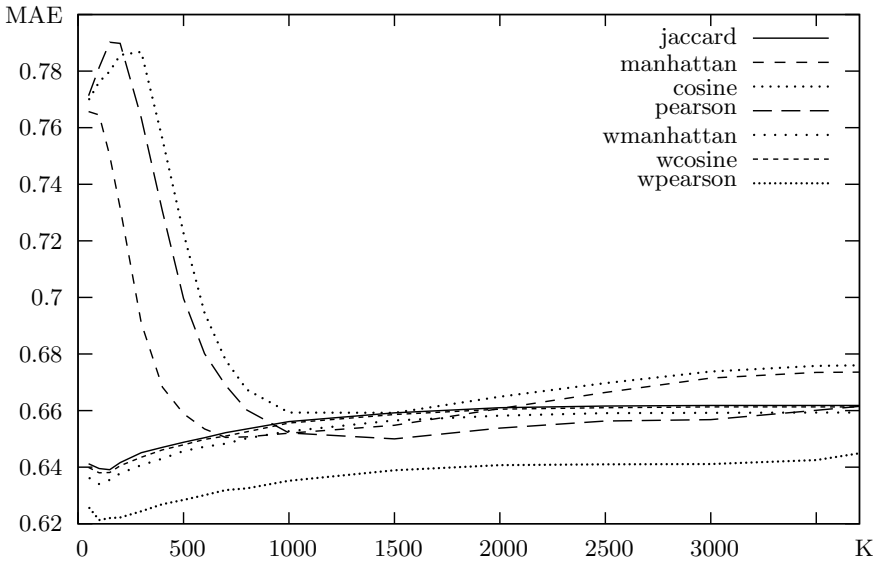

Fig. 1. Comparing MAE on MovieLens when using item-based approaches with different similarity measures and neighbourhood sizes (K).

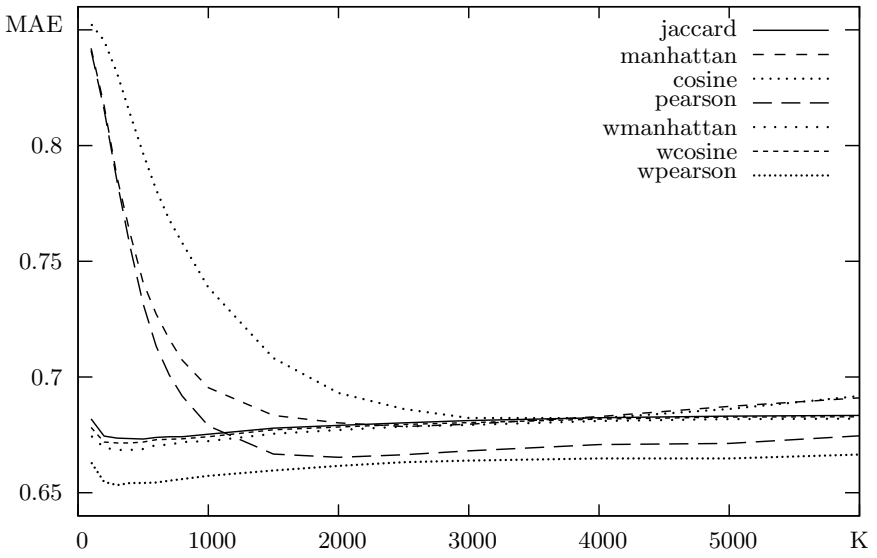

Fig. 2. Comparing MAE on MovieLens when using user-based approaches with different similarity measures and neighbourhood sizes (K).

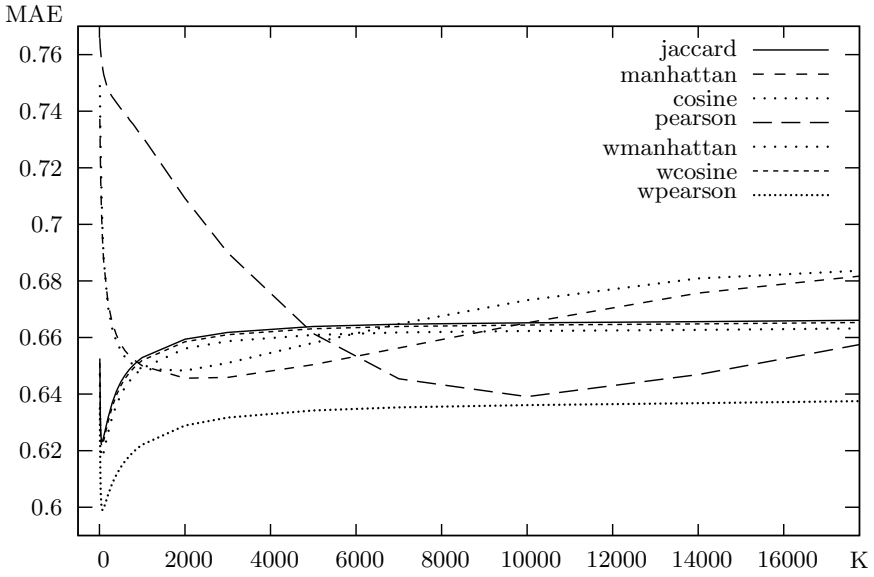

Fig. 3. Comparing MAE on Netflix when using item-based approaches with different similarity measures and neighbourhood sizes (K).

one. On Netflix, considering 70 neighbours leads to the lowest error rate. User-based approaches, however, face scalability issues. It is too expensive to compute the entire user-user matrix. So instead, we begin by running a clustering, and we then consider as potential neighbours only the users that belong to the same cluster. Considering many neighbours improves the results. But a model based on 1000 neighbours, selected starting from a clustering with 90 clusters, needs 9 hours to learn, 28 minuts to predict, and 10Go RAM. The best overall results are reached using an item-based approach. It needs 2 hours and a half to learn the model on Netflix, and 1 minut to produce 10 millions rating predictions. A precision_5 of 0.6216 means that 62.16% of the best rated items are captured by the system and proposed to the users.

Now let us concentrate on item-based approaches and pearson-based similarities. Figure 4 shows the mean number of attributes in common between the nearest item neighbours, depending on the similarity measure used and the number K of nearest neighbours considered. It thus shows that when pearson is used, the nearest neighbours in fact do not share many attributes. They often have only one attribute in common. On the contrary, by using jaccard similarity, the selected neighbours are those that share a maximum number of attributes. Figure 4 shows us that many items share more than 100 attributes on the MovieLens dataset. Jaccard searches to optimise the number of common attributes between vectors, but this may not be the best solution for nearest neighbour selection

	neighbour number	learning time	prediction time	MAE	RMSE	$precision_5$	$precision_4$
default	/	**1 sec.**	**1 sec.**	0.6855	0.9234	0.5451	0.7676
user-based	300	4 min.	3 sec.	0.6533	0.8902	0.5710	0.7810
item-based	**100**	2 min.	1 sec.	**0.6213**	**0.8550**	**0.5864**	**0.7915**

Table 1. Summary of the best results on MovieLens depending on the type of approach.

	neighbour number	learning time	prediction time	MAE	RMSE	$precision_5$	$precision_4$
default	/	**6 sec.**	**15 sec.**	0.6903	0.9236	0.5524	0.7392
user-based	1000	9 h	28 min.	0.6440	0.8811	0.5902	0.7655
item-based	**70**	2 h 30	1 min.	**0.5990**	**0.8436**	**0.6216**	**0.7827**

Table 2. Summary of the best results on Netflix depending on the type of approach.

since the values of the vectors on the shared attributes may differ. Weighted pearson is then an interesting compromise between pearson and jaccard.

Then figures 5 and 6 show the results obtained when raising the similarities to some different power (but still preserving their sign). By doing that, since the measures are comprised between -1 and 1, the influence of the nearest neighbours increases while the influence of the furthest neighbours decreases. We can thus observe that the results improve when raising weighted pearson to some power. On the contrary, the results do not change very much when simple pearson is raised to some power, and even sometimes results are worse. So this shows that our proposed weighting scheme gives meaning to the notion of neighbourhood, since giving more trust to the nearest neighbours improves the results, and considering more and more neighbours does not decrease the results anymore.

Finally, tables 3 and 4 summarise the results obtained with item-based approaches and pearson-based similarities. They compare the performance depending on whether our proposed weighting scheme is used or not. It thus clearly shows the interest of weighting the traditional similarity measures, since all performance measures are improved. Fewer neighbours need to be selected, learning time and prediction time are lower, and the predictive performance are significantly better.

We have also tested other ensemblist similarity measures than jaccard that can be found in [12]: Simple Matching, Russel and Rao, Sokal and Sneath, Rogers and Tanimoto, Dice, and Yule. But none of these did improve the results.

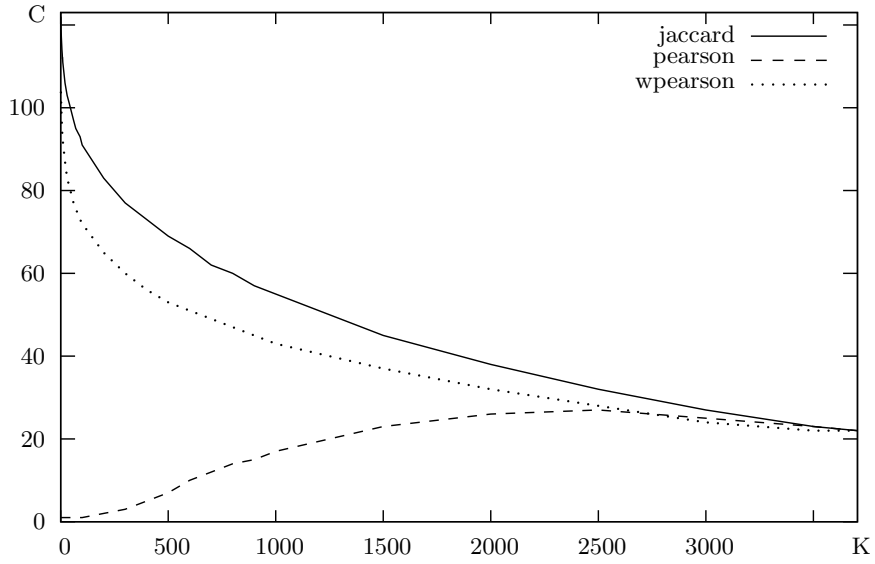

Fig. 4. Comparing the mean number of common attributes (C) between nearest item neighbours on MovieLens when considering different neighbourhood sizes (K).

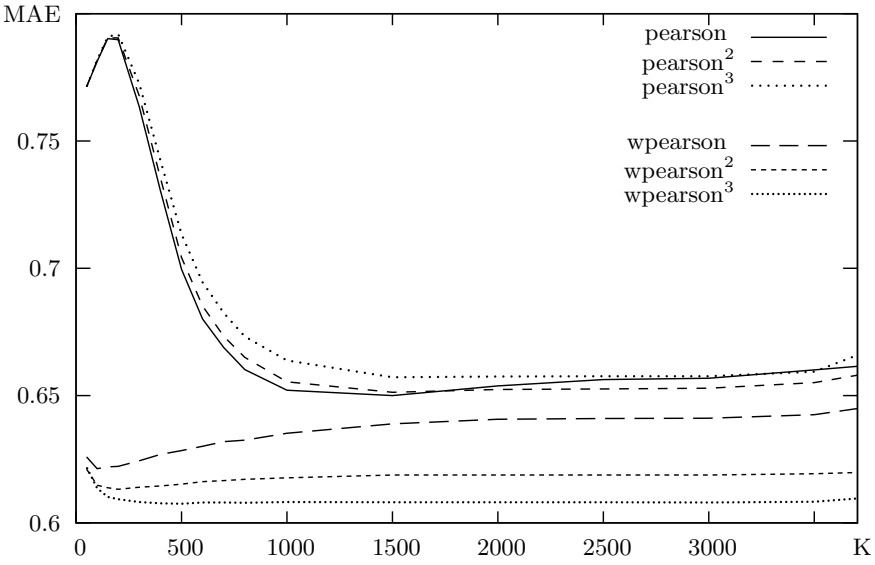

Fig. 5. Comparing MAE on MovieLens when using item-based approaches with different pearson-based similarity measures and neighbourhood sizes (K).

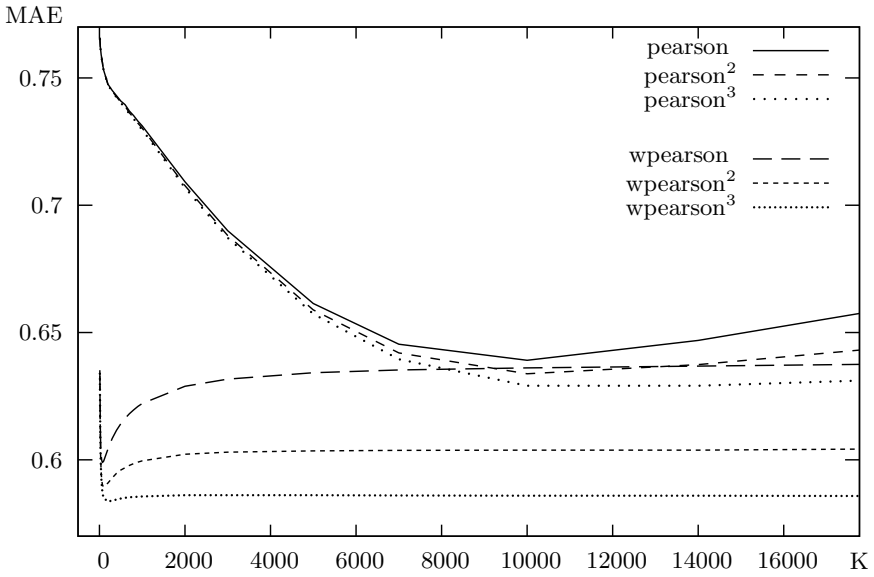

Fig. 6. Comparing MAE on Netflix when using item-based approaches with different pearson-based similarity measures and neighbourhood sizes (K).

	neighbour number	raised power	learning time	prediction time	MAE	RMSE	precision$_5$	precision$_4$
non-weighted	1500	1	3 min.	7 sec.	0.6500	0.8798	0.5703	0.7808
weighted	**500**	**3**	**2 min.**	**3 sec.**	**0.6075**	**0.8404**	**0.5957**	**0.7958**

Table 3. Summary of the best results of item-based approaches on MovieLens depending on the use of our proposed weighting scheme.

	neighbour number	raised power	learning time	prediction time	MAE	RMSE	precision$_5$	precision$_4$
non-weighted	10000	3	7 h	38 min.	0.6291	0.8675	0.6042	0.7712
weighted	**200**	**3**	**2 h 30**	**1 min.**	**0.5836**	**0.8297**	**0.6352**	**0.7881**

Table 4. Summary of the best results of item-based approaches on Netflix depending on the use of our proposed weighting scheme.

4 Conclusion

We have shown in this paper that weighting traditional similarity measures used for collaborative filtering can substantially improve the results of both user- and item-based approaches. More specifically, we have shown that combining pearson and jaccard similarities leads to the best results. This has been verified on two widely-used and independent movie datasets: MovieLens and Netflix.

We have explain that such weighting schemes are useful because traditional similarity measures, such as pearson, have the drawback of considering as neighbours rating vectors that share too few attributes. On the contrary, by using our proposed weighted measures, we have verified that the vectors considered as neighbours share many attributes.

Moreover, we have observed that using these weighted measures allows us to consider fewer neighbours than when non-weighted measures are used. So learning and prediction times of both user- and item-based collaborative filtering approaches are also improved thanks to our new proposition. Finally, we have also observed that raising the weighted pearson similarity to some power still improves the results.

We have also confirmed that item-based approaches outperform user-based ones. Besides their better results, item-based methods have other advantages. Their learning and prediction times are lower, at least for datasets that contain more users than items. They are able to produce relevant predictions as soon as a user has rated one item. Moreover, such models are also appropriate for the navigation in item catalogues even when no information about the current user is available, since it can also present to a user the nearest neighbours of any item she is currently interested in.

Starting from the new similarity measure we have proposed, many approaches for its optimisation may be considered. Content information on items could be used for its improvement [13]. In particular, when an item is associated to very few ratings, then finding its neighbours based on those ratings has little meaning. Instead, using its content description could lead to find more relevant neighbours. The similarity matrix could also be modified by using stochastic perturbations led by cross-validation, or any other optimisation process [14]. Finally, other schemes for combining jaccard and pearson than simple product may also be tested.

Another way to continue improving item-based approaches is to adapt the neighbourhood size parameter to every item independently. Indeed, instead of considering a global parameter fixing the number of neighbours to be considered by each item, we could associate to each item its own parameter. Such parameters could be optimised using cross-validation.

Finally, combining different approaches could also lead to better results. Many ensemble methods could then be implemented in that field [15, 16].

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering **17** (2005) 734–749
2. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Conference on Computer Supported Cooperative Work, ACM (1994) 175–186
3. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: 10th International World Wide Web Conference. (2001)
4. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems **22** (2004) 5–53
5. McNee, S., Riedl, J., Konstan, J.: Being accurate is not enough: How accuracy metrics have hurt recommender systems. In: Extended Abstracts of the 2006 ACM Conference on Human Factors in Computing Systems. (2006)
6. NetflixPrize: http://www.netflixprize.com/ (2006)
7. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating "word of mouth". In: ACM Conference on Human Factors in Computing Systems. Volume 1. (1995) 210–217
8. Karypis, G.: Evaluation of item-based top-N recommendation algorithms. In: 10th International Conference on Information and Knowledge Management. (2001) 247–254
9. Linden, G., Smith, B., York, J.: Amazon.com recommendations: Item-to-item collaborative filtering. IEEE Internet Computing **7** (2003) 76–80
10. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Transactions on Information Systems **22** (2004) 143–177
11. Candillier, L., Meyer, F., Boullé, M.: Comparing state-of-the-art collaborative filtering systems. In Perner, P., ed.: MLDM'2007: 5th International Conference on Machine Learning and Data Mining in Pattern Recognition. Volume LNAI 4571 of LNCS., Leipzig, Germany, Springer Verlag (2007) 548–562
12. Janowitz, M.F.: A combinatorial introduction to cluster analysis. Technical report, Classification Society of North America (2002)
13. Vozalis, M., Margaritis, K.G.: Enhancing collaborative filtering with demographic data: The case of item-based filtering. In: 4th International Conference on Intelligent Systems Design and Applications. (2004) 361–366
14. Bell, R., Koren, Y.: Improved neighborhood-based collaborative filtering. In: ICDM'07: IEEE International Conference on Data Mining, San Jose, California, USA, ACM (2007) 7–14
15. Polikar, R.: Ensemble systems in decision making. IEEE Circuits & Systems Magazine **6** (2006) 21–45
16. Bell, R., Koren, Y., Volinsky, C.: Modeling relationships at multiple scales to improve accuracy of large recommender systems. In: KDD'07: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, ACM (2007) 95–104