Comparing state-of-the-art collaborative filtering systems

Laurent Candillier, Frank Meyer, Marc Boullé
France Telecom R&D Lannion, France
lcandillier@hotmail.com

Abstract. *Collaborative filtering* aims at helping *users* find *items* they should appreciate from huge catalogues. In that field, we can distinguish *user-based*, *item-based* and *model-based* approaches. For each of them, many options play a crucial role for their performances, and in particular the similarity function defined between users or items, the number of neighbors considered for user- or item-based approaches, the number of clusters for model-based approaches using clustering, and the prediction function used.
In this paper, we review the main collaborative filtering methods proposed in the litterature and compare them on the same widely used real dataset called *MovieLens*, and using the same widely used performance measure called *Mean Absolute Error* (MAE). This study thus allows us to highlight the advantages and drawbacks of each approach, and to propose some default options that we think should be used when using a given approach or designing a new one.

1 Introduction

Recommender systems [1] have known a growing interest in the last two decades, since the appearance of the first papers in the mid-1990s [2]. The aim of such systems is to help *users* find products (called more generally *items*) they should appreciate from huge catalogues. To do this, three types of approaches are commonly used:

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

In the first case, the input of the system is a set of ratings of users on sets of items, and the approach used to predict the rating of a given user on a given item is based on the ratings of a set of users who have already rated the given item and whose tastes are similar to the ones of the given user.

In the second case, the item descriptions are used to construct *user thematic profiles* (such as *"like comedy and dislike war"* when items are movies), and the prediction of interest of a user on a given item is based on the similarity between the item description and the user profile. In the third case of hybrid filtering, both information, collaborative and content-based, are used.

In this paper, we focus on the first type of techniques, because it is the most widely considered in the field of recommender systems, and yet many different collaborative filtering approaches are worth to be compared. Besides, in many cases, well-structured item descriptions are hard to get, whereas collecting user ratings on items is easier, yet some real rating datasets are available for tests. We chose the most widely used one, called *MovieLens*, for our study. This dataset contains 1,000,209 ratings collected from 6,040 users on 3,699 items that represent movies. Then two ways for evaluating the performances of a collaborative filtering method can be used [3]:

1. evaluate its error rate in cross-validation,
2. or evaluate user satisfaction in the system.

We focus in this paper on the first approach that is the most widely used one. Many measures can then be used to compare the results of different collaborative filtering methods. The most widely used ones are:

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

The two first measures evaluate the capability of a method to predict if a user will like or dislike an item, whereas the third measure evaluates its capability of providing an ordered list of items that a user should like. So these measures carry different meanings [4]: in the first two cases, the method needs to be able to predict dislike, but there is no need for ordering items, whereas in the third case, the method only focuses on items users will like, but the order in which these items are ranked is important. In this paper, we focus on the MAE, that is the most widely used measure.

The rest of the paper is organized as follows: section 2 presents an overview of the principal approaches for collaborative filtering; we then report in section 3 the results of extensive experiments conducted using various collaborative filtering methods and various alternatives of each, on the *MovieLens* dataset using cross-validation and the MAE measure for comparison; finally, section 4 concludes the paper and proposes some default options that we think should be used when using a given collaborative filtering method, or designing a new one.

2 Collaborative filtering approaches

Let U be a set of users and I a set of items. v_{ui} denotes the rating of user $u \in U$ on item $i \in I$, and $S_u \subseteq I$ stands for the set of items that user u has rated. We assume all ratings are integers ranging from 1 to 5, which is the case for *MovieLens*.

2.1 User-based approaches

For user-based approaches [2], the prediction of rating p_{ai} of user a (active) on item i is computed using the sum of the user mean rating and the weighted sum of deviations from their mean rating of users that have rated item i. More formally, p_{ai} is computed as follows:

$$p_{ai} = \overline{v_a} + \frac{\sum_{\{u \in U | i \in S_u\}} w(a,u) \times (v_{ui} - \overline{v_u})}{\sum_{\{u \in U | i \in S_u\}} |w(a,u)|} \tag{1}$$

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

$$\overline{v_u} = \frac{\sum_{i \in S_u} v_{ui}}{|S_u|} \tag{2}$$

And $w(a,u)$ stands for the similarity between users a and u, computed using *pearson* correlation in [2], that corresponds to the cosine of the users deviation from their mean:

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u} (v_{ai} - \overline{v_a})(v_{ui} - \overline{v_u})}{\sqrt{\sum_{i \in S_a \cap S_u} (v_{ai} - \overline{v_a})^2 \sum_{i \in S_a \cap S_u} (v_{ui} - \overline{v_u})^2}} \tag{3}$$

The influence of this similarity measure in the performances of this approach is very important. So many other measures have been considered in the litterature [5, 6]. Let us introduce two of them:

– simple *cosine*:

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u} v_{ai} \times v_{ui}}{\sqrt{\sum_{i \in S_a \cap S_u} v_{ai}^2 \sum_{i \in S_a \cap S_u} v_{ui}^2}} \tag{4}$$

– and *constraint* pearson correlation, that corresponds to the cosine of users deviation from the mean rating, denoted by M, and equal to 3 for a rating scale ranging from 1 to 5:

$$w(a,u) = \frac{\sum_{i \in S_a \cap S_u} (v_{ai} - M)(v_{ui} - M)}{\sqrt{\sum_{i \in S_a \cap S_u} (v_{ai} - M)^2 \sum_{i \in S_a \cap S_u} (v_{ui} - M)^2}} \tag{5}$$

Finally, a neighborhood for each user can be considered. In such a case, the neighborhood size is then a system parameter that needs to be defined, and only the neighbors of the active user are considered for predictions.

2.2 Model-based approaches

Since predicting the rating of a given user on a given item requires the computation of the similarity between the given user and all its neighbors that have already rated the given item, its execution time may be long for huge datasets. In order to reduce such execution time, model-based approaches have been proposed [7]. The general idea is to derive off-line a model of the data in order to predict on-line ratings as fast as possible.

The first types of models that have been proposed consist in grouping the users using clustering and then predicting the rating of a given user on a given item using only the ratings of the users that belong to the same cluster. Then probabilistic clustering algorithms have been used in order to allow users to belong, at some level, to different groups of users [8, 9]. Hierarchies of clusters have also been proposed, so that if a given cluster of users does not have opinion on a given item, its *super-cluster* can be considered [10].

In such approaches, the choice of the distance measure used to compare users is important. Let us present two widely used of them:

1. normalized *manhattan* distance:

$$dist(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} |v_{ai} - v_{ui}|}{|\{i \in S_a \cap S_u\}|} \tag{6}$$

2. and normalized *euclidian* distance:

$$dist(a,u) = \sqrt{\frac{\sum_{\{i \in S_a \cap S_u\}} (v_{ai} - v_{ui})^2}{|\{i \in S_a \cap S_u\}|}} \tag{7}$$

The number of clusters is also of key importance. In many cases, different numbers of clusters are tested, and the one that led to the lowest error rate in cross-validation is kept. Clusters C_k are then generally represented by their centroid $\overrightarrow{\mu_k}$:

$$\mu_{ki} = \frac{\sum_{\{u \in C_k | i \in S_u\}} v_{ui}}{|\{u \in C_k | i \in S_u\}|} \tag{8}$$

Then the predicted rating of a user to an item can be directly derived from the rating of its nearest centroid, or it can be computed using a sum on the ratings of all centroids, weighted by the distance between the given user and the centroids.

For this study, we implemented four clustering algorithms:

– *K-means*, the well-known full-space clustering algorithm based on the evolution of K centroids that represent the K clusters to be found,
– *Bisecting* K-means [11], based on the recursive use of (K=2)-means, by selecting at each step for next split the cluster that maximizes its inertia,
– *LAC* [12], that is based on K-means and adds a weight to each attribute, depending on the deviation of the cluster members from its mean,
– and *SSC* [13], that is a probabilistic clustering algorithm, based on a mixture of gaussians and the *EM* algorithm.

All these methods need to be run many times with random initial solutions in order to avoid local minimum solutions. We set the default number of runs to 10 in these experiments.

Finally, models based on item associations have also been considered. Bayesian models have been proposed to model dependencies between items [7]. The clustering of items have been studied in [14, 15]. And models based on association rules have been studied in [16, 17].

2.3 Item-based approaches

Then item-based approaches have known a growing interest [18]. Given a similarity measure between items (like cosine or pearson correlation presented earlier for user-based approaches), item-based approaches predict the rating of a given user on a given item using the ratings of the user on the items considered as similar to the target item. In [18], a weighted sum is used to predict the rating of active user a on item i, given $sim(i,j)$ a similarity measure between items:

$$p_{ai} = \frac{\sum_{\{j \in S_a | j \neq i\}} sim(i,j) \times v_{aj}}{\sum_{\{j \in S_a | j \neq i\}} |sim(i,j)|} \tag{9}$$

Two specific similarity measures have been proposed in [19, 20] for item-based collaborative filtering methods:

– *adjusted* cosine, that corresponds to the cosine of items deviation from the user mean rating:

$$sim(i,j) = \frac{\sum_{\{u \in U | i \in S_u \& j \in S_u\}} (v_{ui} - \overline{v_u})(v_{uj} - \overline{v_u})}{\sqrt{\sum_{\{u \in U | i \in S_u \& j \in S_u\}} (v_{ui} - \overline{v_u})^2 \sum_{\{u \in U | i \in S_u \& j \in S_u\}} (v_{uj} - \overline{v_u})^2}} \tag{10}$$

– and a *probabilistic* similarity measure, that corresponds to the co-occurrence frequence of both items i and j, normalized by user frequences in order to enhance the contribution of users who have rated fewer items, and then normalized by the product of the frequences of both concerned items:

$$sim(i,j) = \frac{\sum_{\{u \in U | i \in S_u \& j \in S_u\}} v_{uj} / |S_u|}{|\{u \in U | i \in S_u\}| \times |\{u \in U | j \in S_u\}|} \tag{11}$$

Finally, as for user-based approaches, a neighborhood for each item can be considered. In such a case, the neighborhood size is then a system parameter that needs to be defined, and only the neighbors of the target item are considered for predictions.

2.4 Complementary approaches

Different default prediction techniques can also been considered, in particular when a method is not able to predict any rating, if a user has no rating, if it has no neighbor, if there is no rating on an item or if an item has no neighbor:

– *mean item* rating,
– *mean user* rating,
– *majority item* rating.
– and *majority user* rating.

We also propose an alternative approach where we consider the recommendation problem as a standard classification problem with two input variables, user u and item i, and one output variable, rating r. We apply the standard *Naive Bayes* approach, assuming that users and items are independent conditionally to the ratings. This approach is based on the following *Bayes* rule used to compute the probability of rating r for a given user u on a given item i:

$$P(r|u,i) = \frac{P(r|u) \times P(r|i)}{P(r)} \times \frac{P(u) \times P(i)}{P(u,i)} \tag{12}$$

$P(r|u)$ stands for the probability of rating r for user u, $P(r|i)$ the probability of rating r on item i, and $P(r)$ the global probability of rating r. The last three probabilities $P(u)$, $P(i)$ and $P(u,i)$ can be ignored since they are the same for all users and items. From these probabilities, we then propose three prediction scheemes:

– predict the most *probable* rating, which corresponds to the *Maximum A Posteriori* (MAP) approach:

$$p_{ai} = Argmax_{r=1}^{5} P(r|a,i) \tag{13}$$

– compute the *weighted* sum of ratings, that corresponds to minimizing the expectation of *Mean Squared Error* (MSE):

$$p_{ai} = \sum_{r=1}^{5} r \times P(r|a,i) \tag{14}$$

– or select the rating that *minimizes* the expectation of *Mean Absolute Error* (MAE):

$$p_{ai} = Argmin_{r=1}^{5} \sum_{n=1}^{5} P(n|a,i) \times |r-n| \tag{15}$$

Model-based approaches can be combined with different default approaches, or with any user- or item-based approach. This is done by constructing local models from the different sub-datasets created using clustering.

Finally, since in many real datasets ratings are integer values, we can choose to round the predicted ratings instead of using their real values. Such a process improves the results when MAE is used, but not when RMSE is used.

3 Experiments

3.1 Parameters

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 3), cosine (4), constraint pearson (5), adjusted cosine (10), or probabilistic (11),
– a neighborhood size,
– and how to compute predictions: using a weighted sum of rating values (9), or using a weighted sum of deviations from the mean (1).

For model-based approaches, the following parameters need to be defined:

– the distance measure used: manhattan (6) or euclidian (7),
– the number of clusters,
– how to compute predictions in one cluster: using the mean rating of the cluster members on an item, using another default approach, or using a local user- or item-based approach,
– and how to compute predictions for one user: returning the prediction of its nearest cluster, or the weighted sum of predictions of each cluster.

Finally, in all cases, we can choose to round the results or not. As a default prediction scheeme, if no prediction can be done for a given approach, the global mean item rating is returned, and if the item is not known by the system, then the mean user rating is returned.

3.2 Protocol

We conduct these experiments using *MovieLens* dataset. We divided it into 10 parts in order to perform 10-fold cross-validations, training the chosen model using 9 parts and testing it on the last part. In all experiments, the division into 10 parts of the dataset is always the same, so that all approaches are evaluated under exactly the same conditions.

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 performances of the algorithms:

$$MAE = \frac{1}{|T|} \sum_{(u,i,r) \in T} |p_{ui} - r| \tag{16}$$

$$RMSE = \sqrt{\frac{1}{|T|} \sum_{(u,i,r) \in T} (p_{ui} - r)^2} \tag{17}$$

We also report the time spent for the model construction and for predictions.

3.3 Results

Let us start with the results of the default approaches presented in table 1.

We can thus already observe that the results are better when default ratings are based on item information than when they are based on user information, and that using the mean rating is better than using the majority rating. But default

	MeanItem	MeanUser	MajoItem	MajoUser	ProbablBayes	WeightBayes	MinBayes
MAE(1)	0.7821	0.8286	0.7702	0.8363	0.7159	0.7279	**0.6829**
MAE(2)	0.7501	0.7939	0.7702	0.8363	0.7159	0.6899	**0.6829**
RMSE(1)	0.9791	1.0350	1.0924	1.1991	1.0658	**0.9247**	0.9894
RMSE(2)	1.0182	1.0741	1.0924	1.1991	1.0658	**0.9684**	0.9894

Table 1. Default approaches results measured using MAE and RMSE, when rounding (2) or not (1) the predicted ratings.

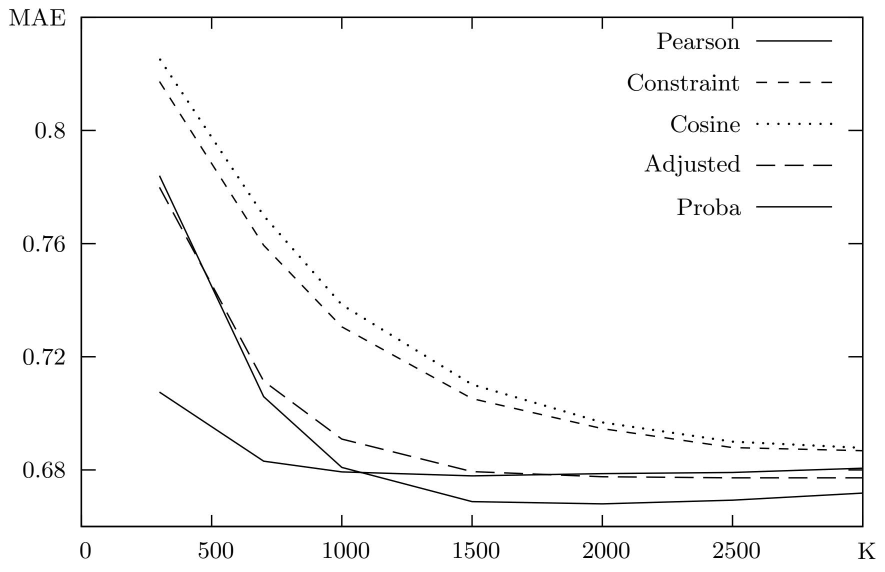
ratings using Bayes models lead to much better results. For such approaches, the MAE is minimized with the MinBayes scheeme (equation 15), but the RMSE is minimized with the WeightBayes sheeme (14). Rounding the predicted ratings improves the results when the MAE is used, but not when the RMSE is used. These two observations confirm the theory. In the following, we only report results using MAE and after the predicted ratings have been rounded. Figure 1 reports such results using different user-based approaches.

We can thus observe that the results are improved when many neighbors are considered. But of course the execution time is higher when more neighbors are used. The similarity measure that leads to the best results is pearson, according to figure 1(a). Predicting using weighted sum of deviations from the mean leads to better results than predicting using simple weighted sum according to figure 1(b). Rounding the predicted ratings improved the MAE from 2 to 4 percent. Figure 2 then reports the results using item-based approaches.

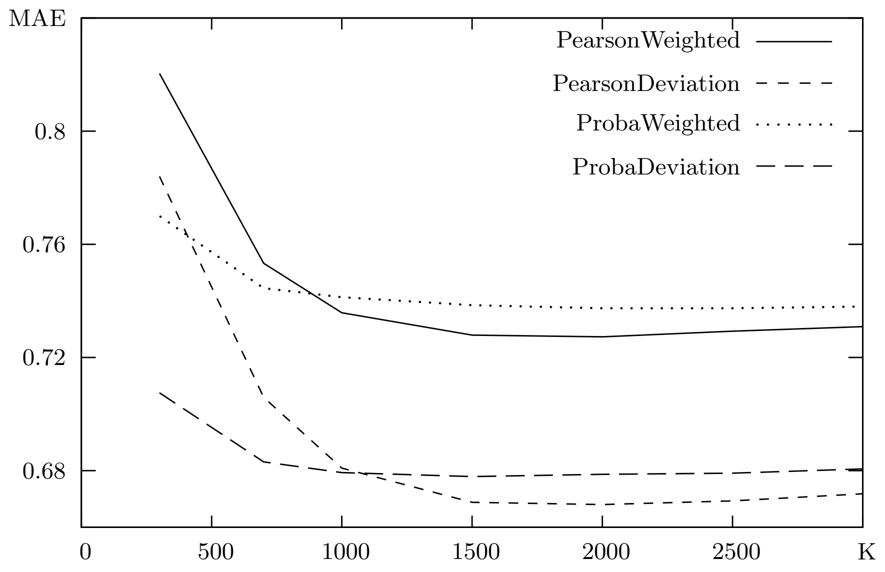
We observe again from figure 2(b) that predicting using weighted sum of deviations from the mean leads to a lower MAE than predicting using simple weighted sum, no matter which similarity measure is used. But in that case, considering too much neighbors degrades the results and the probabilistic similarity leads to the lowest MAE, according to figure 2(a). Rounding the predicted ratings improved the MAE from 3 to 4 percent. Finally, figure 3 reports the results obtained using Kmeans-based approaches.

According to figure 3(a), we see that there is not a high difference in using manhattan or euclidian distance, although euclidian distance leads to slightly better results. In both cases, the optimal number of clusters is 6. On the contrary, predicting using MinBayes leads to better results than when MeanItem rating is used, according to figure 3(b), and in that case, the optimal number of clusters is 4. Those reported results concern predictions based on the nearest cluster rather than based on a weighted sum of predictions of each cluster because that first scheeme led to better results. Figure 4 presents results using other clustering algorithms than K-means, and shows that K-means outperforms LAC and SSC, but that Bisecting K-means can lead to better results when more clusters are considered.

Finally, table 2 summarizes the results of the best of each approach, including execution time.

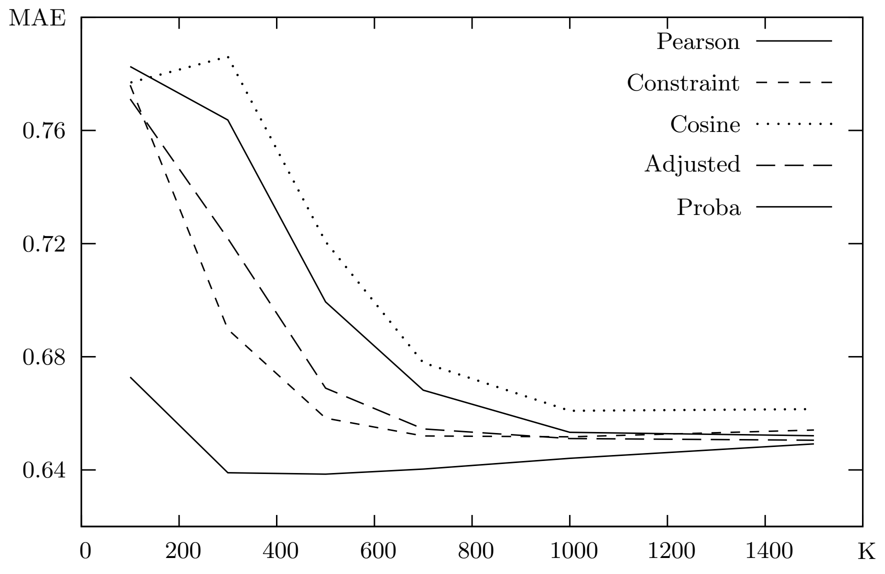


(a) Comparing similarity measures.

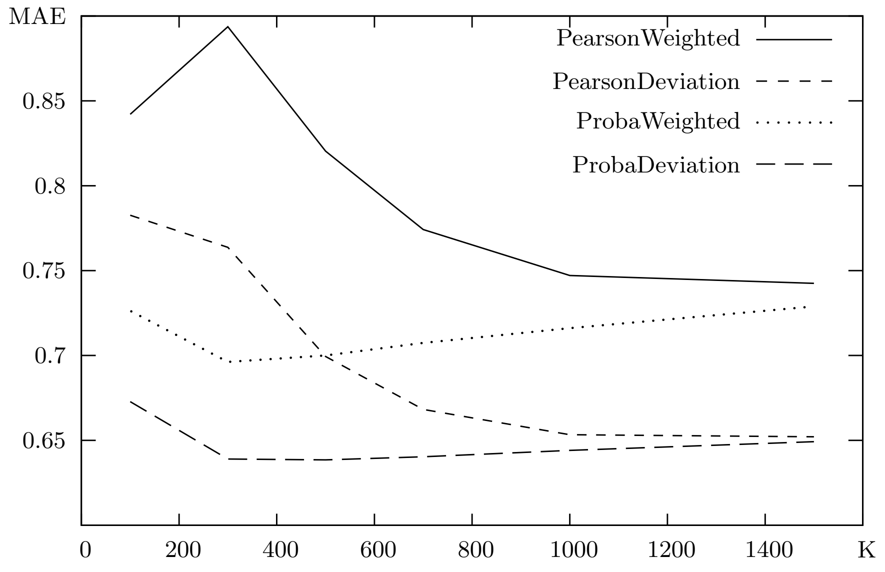


(b) Comparing prediction scheeme.

Fig. 1. User-based approaches results using different neighborhood sizes (K), similarity measures and prediction scheemes.

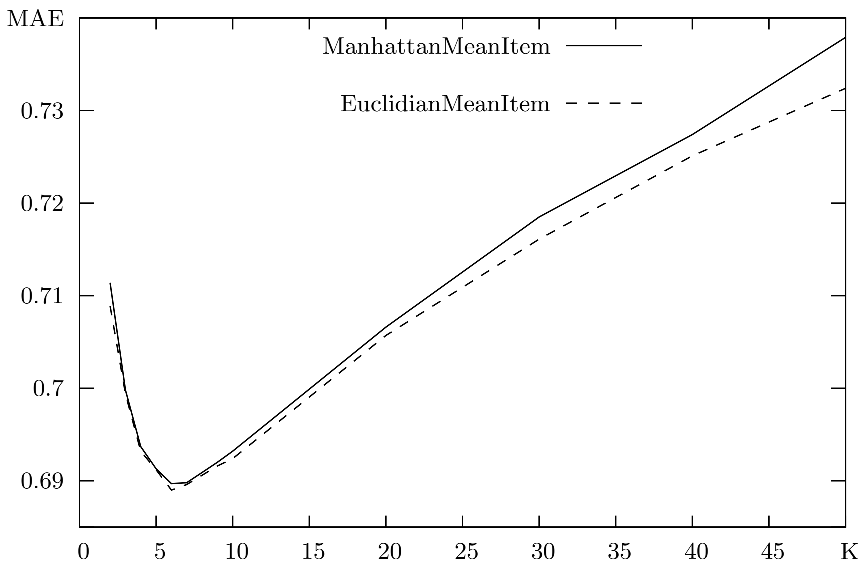


(a) Comparing similarity measures.

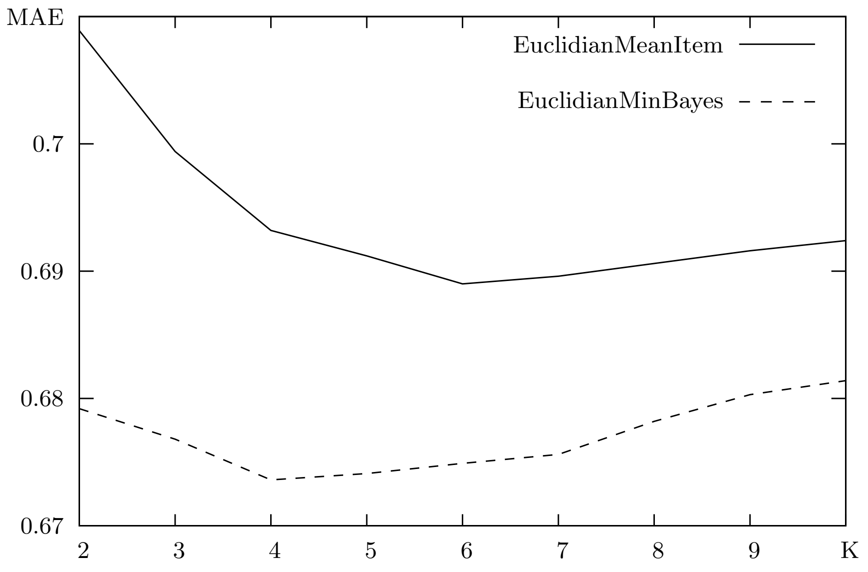


(b) Comparing prediction scheeme.

Fig. 2. Item-based approaches results using different neighborhood sizes (K), similarity measures and prediction scheemes.



(a) Manhattan versus euclidian distance.



(b) MeanItem versus MinBayes prediction.

Fig. 3. Model-based approaches results using different numbers of clusters.

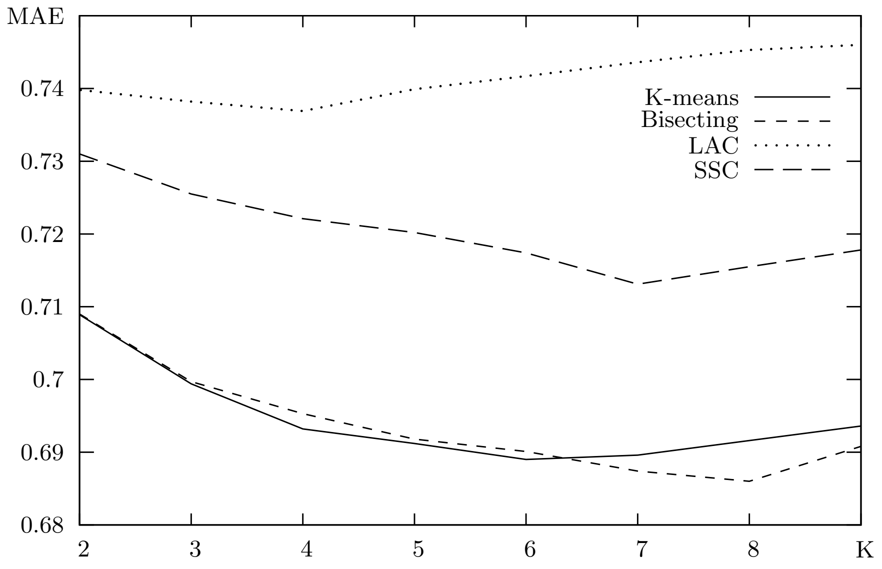


Fig. 4. Results of model-based approaches using different clustering algorithms.

	BestDefault	BestUserBased	BestItemBased	BestModelBased
model construction time (in sec.)	**1**	730	170	254
prediction time (in sec.)	**1**	31	3	24
MAE	0.6829	0.6688	**0.6382**	0.6736

Table 2. Summary of the best approaches.

The *BestDefault* is MinBayes: the default approach based on Bayes rule minimizing MAE (equation 15). The *BestUserBased* is the user-based approach based on pearson similarity (3) and 1500 neighbors. The *BestItemBased* is the item-based approach based on probabilistic similarity (11) and 400 neighbors. Both use predictions using weighted sum of deviations from the mean (1). Finally, the *BestModelBased* is the model-based approach using K-means with euclidian distance (7), 4 clusters and predictions scheeme based on the nearest cluster and Bayes model minimizing MAE.

The best overall results are reached by the best item-based approach. It needs 170 seconds to construct the model and 3 seconds to predict 100,021 ratings. Then the best user-based approach has slightly lower MAE than model-based or default approaches, but for a model construction time of 730 seconds and prediction time of 31 seconds. On the other side, the best default approach only needs 2 seconds for both model construction and predictions, for a difference in MAE of only 0.0141.

Finally, we also tested the use of local item-based models constructed on the different user groups identified by clustering, but such approach degrades the results of a global item-based approach.

4 Conclusion

According to our first results on default approaches, it seems that using Bayes model for default predictions is relevant, since it has reasonable error rate for very low execution time. Besides, another important advantage of such a technique is that it is easily updatable since it is incremental, whereas the other approaches need to relearn their entire model in order to take into account new data.

For all experiments, rounding the predicted ratings led to an improvement ranging from 2 to 4 percent of the MAE. Besides, rounding ratings is natural in practice, since real users generally prefer rating scales based on natural numbers than on real numbers.

Computing predictions using weighted sum of deviations from the mean also led to better results than using simple weighted sum for both user- and item-based approaches. The lowest error rates were reached using pearson similarity for user-based approaches and probabilistic similarity for item-based approaches.

Using Bayes default approach in order to predict ratings inside a given cluster leads to better results than when the mean rating of the cluster members is used. Considering the prediction of the nearest cluster is better than computing a weighted sum of the predictions of each cluster. Finally, K-means had better results than more sophisticated algorithms like LAC or SSC.

More generally, item-based approaches seem the bests in our experiments. But these results need to be taken with precaution. Indeed, although in many cases the number of users is much more important than the number of items, in cases where there are more items than users, user-based approaches could lead to better results. On the same way, if there are some demographic information on users, results of user-based approaches can be improved [21]. On the other side, if some content information on items are available, results of item-based approaches can also be improved [22].

Besides their very good results, item-based approaches have other advantages: they seem to need fewer neighbors than user-based approaches, and 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 neighbors of any item he is currently interested in.

For future work, it seems now interesting to study how these methods can be adapted to scale well when faced with huge datasets. The dataset provided by Netflix [23], a popular online movie rental service, can be used for such tests since it contains 100,480,507 movie ratings from 480,189 users on 17,770 movies.

In that field, let us cite [24] that proposed a user selection scheeme for user-based approaches, [11] that proposed to create *super-users* by running a user-based approach considering as users the centroids found using a bisecting K-means clustering algorithm, [25] that proposed to use *Principal Components Analysis* (PCA) or [26] that proposed to use *Singular Value Decomposition* (SVD) in order to reduce the initial rating matrix size.

Unfortunately, such dimensionality reduction techniques then prevent us from presenting understandable results to the users because of the rating matrix transformation. So instead, we think it is interesting to study how bagging [27] could be used in collaborative filtering, and if using local item-based approaches in each cluster found using K-means still fails with huge datasets such as Netflix's one.

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. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems **22** (2004) 5–53
4. 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)
5. 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
6. Weng, J., Miao, C., Goh, A., Shen, Z., Gay, R.: Trust-based agent community for collaborative recommendation. In: 5th International Joint Conference on Autonomous Agents and Multiagent Systems. (2006)
7. Breese, J., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufman (1998) 43–52

8. Pennock, D., Horvitz, E., Lawrence, S., Giles, C.L.: Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In: 16th Conference on Uncertainty in Artificial Intelligence. (2000) 473–480
9. Kleinberg, J., Sandler, M.: Using mixture models for collaborative filtering. In: 36th ACM Symposium on Theory Of Computing, ACM Press (2004) 569–578
10. Kelleher, J., Bridge, D.: Rectree centroid : An accurate, scalable collaborative recommender. In Cunningham, P., Fernando, T., Vogel, C., eds.: 14th Irish Conference on Artificial Intelligence and Cognitive Science. (2003) 89–94
11. Rashid, A.M., Lam, S.K., Karypis, G., Riedl, J.: ClustKNN: A highly scalable hybrid model- & memory-based CF algorithm. In: KDD Workshop on Web Mining and Web Usage Analysis. (2006)
12. Domeniconi, C., Papadopoulos, D., Gunopulos, D., Ma, S.: Subspace clustering of high dimensional data. In: SIAM International Conference on Data Mining. (2004)
13. Candillier, L., Tellier, I., Torre, F., Bousquet, O.: SSC : Statistical Subspace Clustering. In Perner, P., Imiya, A., eds.: 4th International Conference on Machine Learning and Data Mining in Pattern Recognition (MLDM'2005). Volume LNAI 3587 of LNCS., Leipzig, Germany, Springer Verlag (2005) 100–109
14. Ungar, L., Foster, D.: Clustering methods for collaborative filtering. In: Workshop on Recommendation Systems, AAAI Press (1998)
15. O'Conner, M., Herlocker, J.: Clustering items for collaborative filtering. In: ACM SIGIR Workshop on Recommender Systems. (1999)
16. Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.: Analysis of recommendation algorithms for e-commerce. In: ACM Conference on Electronic Commerce. (2000) 158–167
17. Lin, W., Alvarez, S., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. In: Data Mining and Knowledge Discovery. Volume 6. (2002) 83–105
18. Karypis, G.: Evaluation of item-based top-N recommendation algorithms. In: 10th International Conference on Information and Knowledge Management. (2001) 247–254
19. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: 10th International World Wide Web Conference. (2001)
20. Deshpande, M., Karypis, G.: Item-based top-N recommendation algorithms. ACM Transactions on Information Systems **22** (2004) 143–177
21. Pazzani, M.J.: A framework for collaborative, content-based and demographic filtering. Artificial Intelligence Review **13** (1999) 393–408
22. 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
23. NetflixPrize: http://www.netflixprize.com/ (2006)
24. Yu, K., Xu, X., Tao, J., Ester, M., Kriegel, H.: Instance selection techniques for memory-based collaborative filtering. In: SIAM Data Mining. (2002)
25. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval **4** (2001) 133–151
26. Vozalis, M., Margaritis, K.: Applying SVD on item-based filtering. In: 5th International Conference on Intelligent Systems Design and Applications. (2005) 464–469
27. Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140