Expert Systems with Applications 39 (2012) 11511–11516

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Similarity of users' (content-based) preference models for Collaborative filtering in few ratings scenario

Alan Eckhardt

Department of Software Engineering, Charles University in Prague, Prague, Czech Republic

ARTICLE INFO

Keywords:
Collaborative filtering
Preference learning
Machine learning

ABSTRACT

Collaborative filtering is an efficient way to find best objects to recommend. This technique is particularly useful when there is a lot of users that rated a lot of objects. In this paper, we propose a method that improve the Collaborative filtering in situations, where the number of ratings or users is small. The proposed approach is experimentally evaluated on real datasets with very convincing results.

1. Introduction

Collaborative filtering has gain great popularity in past years, mainly thanks to the Netflix competition (http://www.netflixprize.com). In this competition, ratings of movies were provided to the competitors, whose task was to predict the ratings of unrated movies for all users. The idea behind Collaborative filtering is to find similar users that had already rated the movie in question, and to analyze the ratings of the movie by these users.

The other approach is content-based filtering. As the name suggests, it is based on knowing what properties the movie should have in order to be preferred. This approach is purely user specific – the information about other users does not come into play. Having the ratings of the objects, the attributes of the objects and trying to find how the attributes influences the rating, the content-based filtering task is similar to machine learning. The key aspect of the method used for content-based filtering is to be transparent, to provide clear information about the preferences. E.g. considering multilayer perceptron, it is not a transparent model at all – the preferences somehow stored in the particular neurons. It is not possible to see, if e.g. movies by a certain director are preferred or not.

In this paper, we will describe previously proposed use model that is capable of providing such clear information about preferences. We will use this model as user similarity measure, instead of traditional ratings-based similarity. In Section 2 is an introduction to Collaborative filtering field, then in Section 3 is the main contribution, the description of computing the similarity of users using their user models. Finally, in Section 5 are experiments and in Section 6 are concluding remarks.

2. Collaborative filtering and related work

Collaborative filtering was proposed in early 90s in Goldberg, Nichols, Oki, and Terry (1992) and further developed. One of the well-known systems using Collaborative filtering is GroupLens Rashid et al. (2002). Collaborative filtering was also described for example in Shardanand and Maes (1995) and Ko and Lee (2002).

Collaborative filtering is based on the idea of similarity of users. When we want to know how user $u_1 \in U$ will like object o, one way is to look how other people liked o. `Amazon.com` succeeds in describing this approach in one sentence "Customers Who Bought This Item Also Bought...".

The better way is to restrict only to those users that are similar to u_1. The similarity of users may be computed in various ways, the most common is the similarity of ratings of objects other than o. This approach is rather accurate, but suffers from a big drawback – there has to be a lot of users and each of them must have done a lot of ratings. Only if there is enough data from users, the computation of similarity gives reliable results.

Other possibility is to compute the similarity of user profiles – e.g. find managers, from 25 to 30, divorced, with interest in psychology and computer science. There is a hidden assumption that similarity in profile implies similarity in preferences, which may not be always true.

The other way of estimating user preference of an item o is to use *item similarity* rather than user similarity. In this approach, we find items similar to o and find the user's preference over these items. The similarity of items is again computed using ratings – if many users rated items o and p with the same rating, these two items are similar. The advantage of this approach is that item similarity does not change in time, but user similarity can. However, with many items and few users, this approach is less useful than user similarity.

Collaborative filtering is necessary when there are no or very few attributes in the data. For example, in Goldberg, Roeder, Gupta,

E-mail address: eckhardt@ksi.mff.cuni.cz

0957-4174/$ - see front matter © 2012 Published by Elsevier Ltd.
http://dx.doi.org/10.1016/j.eswa.2012.01.177

and Perkins (2001) is studied a database of ratings of jokes. To say what a good joke is in strict logic speaking is rather difficult, if not impossible. But using Collaborative filtering, the taste of other users can be used without having to know what the actual preferences of the user are.

The lack of the knowledge of why the user prefers an object is on the other hand a substantial drawback. The motive of the user can be used when creating the user interface, as proposed in Václav, Eckhardt, and Vojtáš (2010).

2.1. Combining content based and Collaborative filtering

There are works that propose ways to combine both filtering techniques – collaborative and content-based. In Claypool et al. (1999) is proposed the most straightforward method – to obtain both collaborative and content-based recommendations and to do a weighted average of both.

In Pazzani (1999) is proposed a similar method to ours – using the content based profile for estimation of user similarity. However, only trivial profiles are used. In Basilico and Hofmann (2004) is proposed a unified view on content and Collaborative filtering.

The lack of actual ratings is addressed in Lee, Park, and Park (2007) and Melville, Mooney, and Nagarajan (2002). In Lee et al. (2007) was proposed using also implicit feedback from the user to fill the rating matrix. Another approach is adopted by Melville et al. (2002), where content model is used to estimate ratings of all objects for all user. Collaborative filtering then uses the matrix with all ratings. This approach addresses the scarcity problem, which occur when there are many objects and not enough users.

The possibility to use the item features and user features for Collaborative filtering was studied in Agarwal and Chen (2009). The proposed approach combines both sources of features and reflects the amount of information about the user.

3. Similarity of users' preference models for Collaborative filtering

3.1. User model

Our user preference model is based on the idea of content based user preference model. The user preferences depend on the value of attributes of the objects, some of the values are preferred and some are not. The overall preference of an object is a combination of preference degrees of the attribute values. Because of the division of the preference to two levels, we will note this preference model as "two step model". It was described in detail in Eckhardt, Pokorný, and Vojtáš (2007).

In the first step, which we call local preferences, every attribute value of object o is normalised and ordered using a fuzzy set $f_i^u : D_{A_i} \to [0,1]$, where i is the index of the attribute and u is the user for who this function applies. We will also use name of the attribute instead of the index; e.g. $f_{Display}$. These fuzzy sets are also called objectives or preferences over attributes. With this transformation, the original space of objects' attributes $\mathcal{X} \subseteq \prod_{i=1}^{N} D_{A_i}$ is transformed into $\mathcal{X}' \subseteq [0,1]^N$. Moreover, we know that the object with transformed attribute values equal to $[1,\ldots,1]$ is the most preferred object. It probably does not exist in the real world, though. On the other side, the object with values $[0,\ldots,0]$ is the least preferred, which is more probable to be found in reality.

Examples of f_i for numerical attributes are in Fig. 1. Note that every user has different slope of the preference. For nominal attributes, pairs (value, rating) are learnt for every attribute value of the attribute, for example colour.

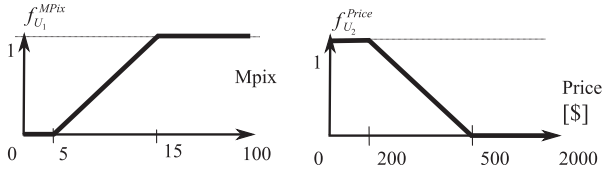

Fig. 1. Examples of preferences for numerical attributes.

In the second step, called global preferences, the normalised attributes are aggregated into the overall score of the object using an aggregation function $@ : \mathcal{X}^{prime} \to [0,1]$. Aggregation functions are also often called utility functions in economy Clemen (1996).

Aggregation function may have different forms; one of the most common is a weighted average, as in the following formula for notebooks:

$$@(o) = \frac{(2 * f_{Price}(o) + 1 * f_{Display}(o) + 1 * f_{RAM}(o))}{4}$$

3.2. Similarity of user models

In this section, we will use the user preference model as the base for computing similarity of users for Collaborative filtering. Since the user model is divided to two steps, the similarity is computed on both levels – local and global preferences. We have proposed a method Statistical Eckhardt (2007), Eckhardt and Vojtáš (2008) for learning such a preference model from ratings. The preference model consists of linear functions for normalization of numerical attributes, "Representants" for nominal attributes and weighted average as aggregation function.

For numerical attributes, the similarity is estimated as the similarity of linear functions $f^{u_1}(x) = \alpha_1 * x + \beta_1$, $f^{u_2}(x) = \alpha_2 * x + \beta_2$. More weight is paid to α parameter, as the ordering of the domain matters more then the offset. The actual weight (3) of α parameter was determined arbitrarily.

$$s(u_1,u_2)_{Price} = 1 - \frac{3 * |\alpha_1 - \alpha_2| + |\beta_1 - \beta_2|}{3 * max(\alpha_1,\alpha_2) + max(\beta_1,\beta_2)}$$

For nominal attributes, the similarity is the similarity of the preferences of common values. E.g. if both users prefer black colour ($f_{Colour}^{u_1}(black) = 0.9$ and $f_{Colour}^{u_2}(black) = 0.85$), they are similar. Essentially, we compute the average difference of preference of common values:

$$s(u_1,u_2)_{Colour} = 1 - \frac{\sum_{a \in Colour} |f_{Colour}^{u_1}(a) - f_{Colour}^{u_2}(a)|}{|Colour| * m},$$

where $m = max(f_{Colour}^{u_1}(a), f_{Colour}^{u_2}(a) | \forall a \in Colour)$ There may be the case that they have no value in common in which case the similarity is 0.

For global preferences, the similarity of two user preference models is the similarity of the weights of the weighted average. Using this similarity, we are able to find the nearest neighbours of a given user u.

$$s(u_1,u_2)_{Global} = \frac{\sum_{i=1,\ldots,N} |w_i^{u_1} - w_i^{u_2}|}{N * max(w_i^{u_1}, w_i^{u_2} | \forall i = 1,\ldots,N)}$$

We will denote this method as "StatColl" meaning Statistical for Collaborative filtering. There will be three flavours of StatColl – one that uses only the ratings of other users, second that uses also the ratings predicted by the user model (i.e. the method Statistical) and the third that uses ratings predicted by the user model also of other users.

The reference will be Taste framework (http://taste.sourceforge.net/old.html) with user correlation method. This is one of the basic approaches for Collaborative filtering, described in

Table 1
Different kinds of Collaborative filtering.

Name of the method	How it computes
StatColl	Weighted average ratings of k nearest users that rated object o
StatColl + user	Weighted average ratings of k nearest users that rated object o and the rating predicted by Statistical
StatColl + others	Weighted average ratings of k nearest users - if they rated the object o, use that rating, otherwise use Statistical to get the predicted rating for o
Collaborative filtering	Uses Taste to predict the rating using k nearest users
Statistical	Uses Statistical to predict the rating of o, no information about other users used

Section 2. Taste will be referred to plainly as "Collaborative filtering". As we concentrate on the case with a small number of users and a small number of ratings, user correlation will be more appropriate for this case.

The main difference and advantage of StatColl over Taste is that the similarity of two users u_1 and u_2 can be estimated in all cases. Taste is able to estimate the similarity only if both users have rated at least one common object. If the set of rated object is disjoint, there is no way of estimation of similarity (when we do not take into account the similarity of objects). This fact results in much more unpredicted objects for Taste, which is confirmed in experiments in Section 5.1.

Table 1 describes the methods for evaluating an unknown object o by the user u. The basic division is whether the preference model is used to evaluate o or not. There is a possibility to use the user's preference model or even to use the preference model of other users that are similar to u.

4. Experiment settings

4.1. Error measures

We will use the following notation: $r(o)$ is the user's rating of object o, $\hat{r}(o)$ is the rating predicted by the method.

Before the list of error measures studied, it is necessary to point out one issue, which came out during experiments. Collaborative filtering for small training data is unable to predict the rating for some of the given objects. The number of evaluated objects influences other error measures and represents an error measure itself. The more objects the method can evaluate, the better. When the method is able to evaluate only a small amount of objects, it is quite useless, because we are limited to the evaluated objects when making the recommendation. For each error measure, we mention how it copes with the unevaluated objects.

The first error measure is prevalent in machine learning. Root mean squared error is computed as follows:

$$RMSE(\hat{r}) = \sqrt{\frac{\sum_{o\in\mathcal{X}}(\hat{r}(o) - r(o))^2}{|\mathcal{X}|}}$$

RMSE is a good estimator of method's ability to correctly capture the preference of a particular object. But we are more interested in the correct *ordering* of objects.

RMSE does not cope with the unevaluated objects. It is measured only for the evaluated objects.

Tau coefficient expresses the similarity of two ordered lists L_1, L_2. In our case, the first list L_1 is ordered according to the user's rating and the second list L_2 according to the rating estimated by the method, in decreasing order, so that the most preferred objects are on the top of the lists. In the most simple case, the lists consist only of IDs of objects. In that case, the tau coefficient is computed according to the number of concordant pairs. A pair of objects o, p is concordant, if either o is before p in both lists, or p is before o in both lists. A pair that is not concordant, is discordant. Then, tau coefficient can be computed like this:

$$\tau(L_1, L_2) = \frac{n_c - n_d}{1/2 * n * (n-1)},$$

where n_c is the number of concordant pairs and n_d is the number of discordant pairs. δ in following formula stands for Kronecker delta – $\delta(condition) = 1$ if *condition* is true, 0 elsewhere.

$$n_c = \sum_{o,p\in\mathcal{X}} \delta(sgn(L_1(o) - L_1(p)) = sgn(L_2(o) - L_2(p)))$$

Tau coefficient reflects the unevaluated objects. The more unevaluated object the method has, the lower is the tau coefficient. For large amounts of unevaluated objects, tau coefficient is significantly decreased. This is obvious in Fig. 3.

Finally, two more "error" measures will be studied. The first is the number of evaluated objects. The less the number is, the worse the method is. It may happen that the method is not able to evaluate an object, because it has not enough information. This will often be the case for Collaborative filtering. The other measure is the time required for building the method from the data. The higher the time is, the longer it takes for the method to build itself. Here, the collaborative filtering will be very good.

4.2. Netflix movie ratings dataset

For testing of Collaborative filtering, we have used the well known Netflix dataset (http://www.netflixprize.com) and merged the movies with data from IMDb (http://www.imdb.com). As the movie name is not unique, the resulting dataset contains less ratings than original Netflix set (90 217 939 against original 100 480 507). We have managed to connect 12 031 out of 17 770 movies in Netflix. Since we are concentrating on the case when there are few ratings from the users, we have restricted the set to those users with around 200 ratings. We get a random set of 1 000 users and tested our approach on that set.

We have restricted the dataset to only 1000 users because we are not concentrating on the problems arising out of enormous data. On the contrary, we are interested in results coming from using only a small training set.

4.3. Sushi

Sushi dataset Kamishima (2003) consists of ratings done by 5000 users; each user rated 10 sushi out of 100. The training size was limited from 3 to 9. For Tau coefficient and monotonicity violation, we omit the training size 9, because it leaves no place for comparing two objects from testing set (the testing set would consist of only one object). The attributes of Sushi dataset are in Table 2.

5. Experiments results

5.1. Netflix dataset

We have used Netflix dataset (described in Section 5.1) for testing of Collaborative filtering. Both Collaborative filtering and all StatColl were used with 100 nearest neighbours.

Table 2
Attributes of Sushi dataset.

Attribute name	Type
Name	Nominal
Style	Nominal
Major group	Nominal
Minor group	Nominal
Heaviness	Numerical
Frequentlyusereats	Numerical
Price	Numerical
Frequently sushi sold	Numerical

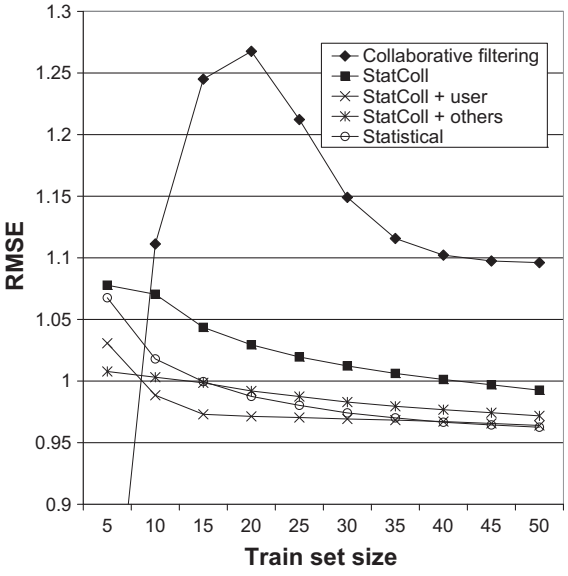

Fig. 2. RMSE for Netflix dataset.

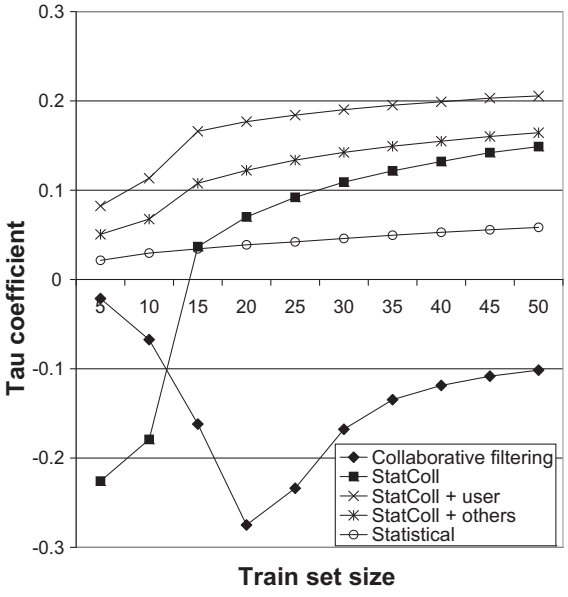

Fig. 3. Tau rank coefficient for Netflix dataset.

For RMSE (Fig. 2) the best method is StatColl + user ($p < 0.001$). Then follows all other StatColls and Statistical, Collaborative filtering is the last ($p < 0.001$ for comparison of Collaborative against other methods). Note that for the training set = 5, Collaborative filtering has very low RMSE. This is because it has very low number

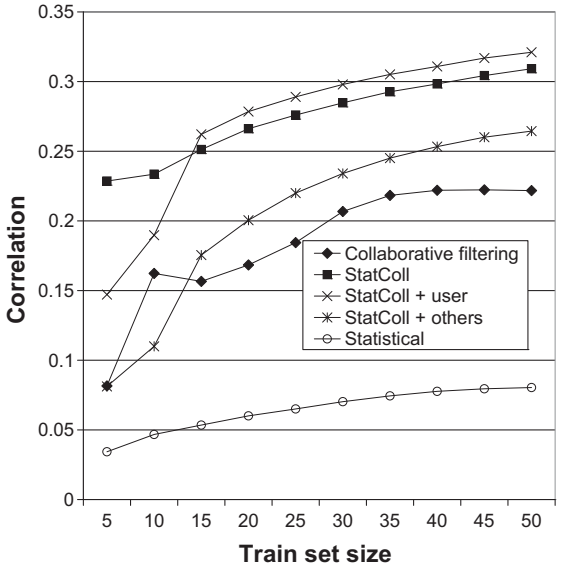

Fig. 4. Correlation for Netflix dataset.

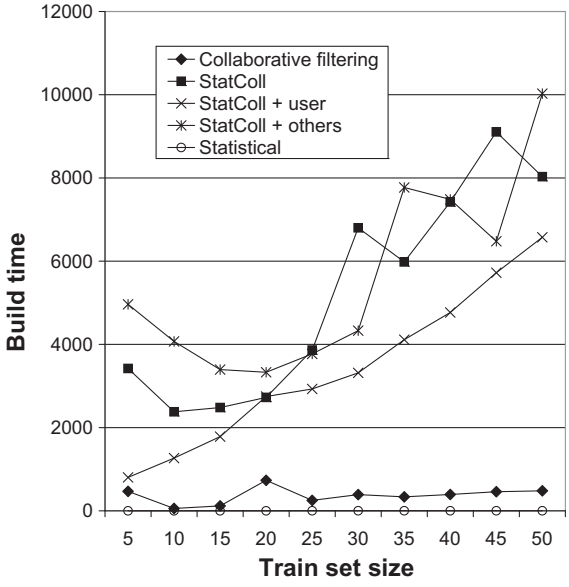

Fig. 5. Build time for Netflix dataset.

of evaluated objects. Actually, it managed to rate only 2.12 objects in average, out of the 195 objects in the testing set.

For Tau coefficient (Fig. 3), StatColl + user is again the best, followed by StatColl + others. StatColl is the third, because it has some unpredicted objects for lower training sets, which penalises it for Tau coefficient. Statistical is the fourth. Collaborative filtering is clearly the worst ($p < 0.001$) because it has many unpredicted objects. This fact penalises its performance for tau coefficient.

It is interesting to observe the behaviour of Collaborative Filtering. For training set size = 5, tau coefficient is about 0. With increasing training set size, tau coefficient decreases. This is counterintuitive, because all other methods improve the performance with increase in training set size. The interpretation for this is that Collaborative filtering for smallest training set sizes evaluate only a very small amount of objects, but it evaluates quite well. With the increase of evaluated object, the precision decreases. Then for training set size larger than 25, there is enough data for Collaborative filtering to improve the prediction and the tau coefficient

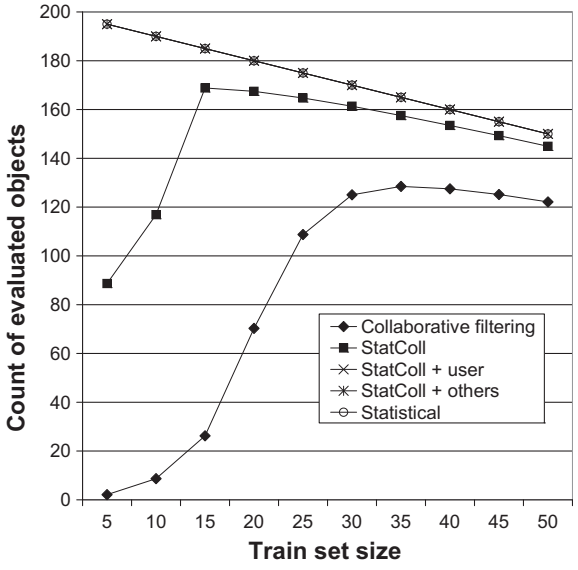

Fig. 6. Count of evaluated objects for Netflix dataset.

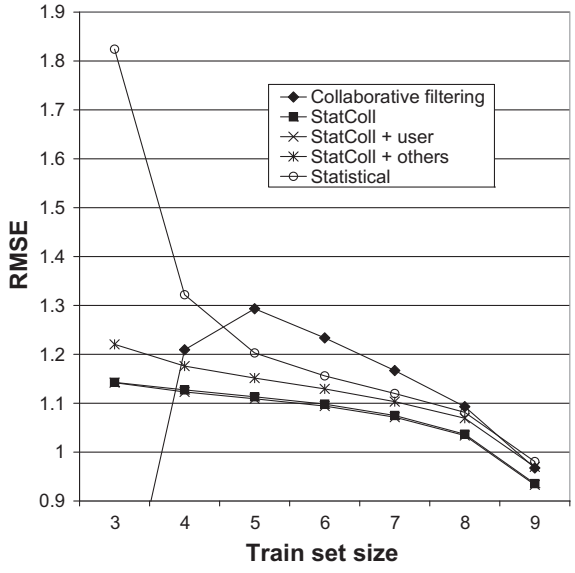

Fig. 7. RMSE for Sushi dataset.

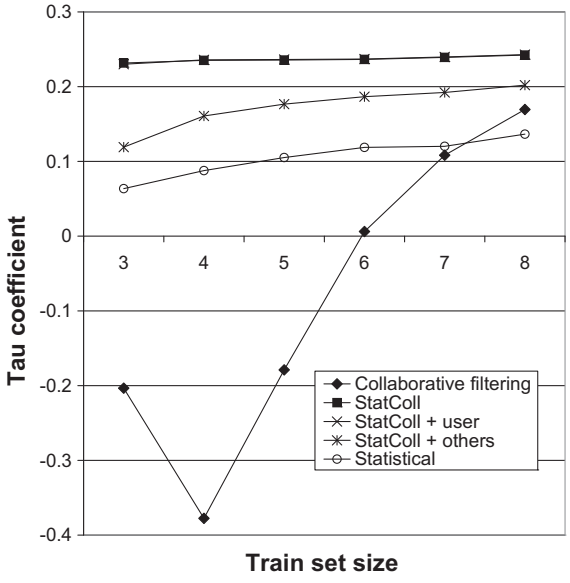

Fig. 8. Tau coefficient for Sushi dataset.

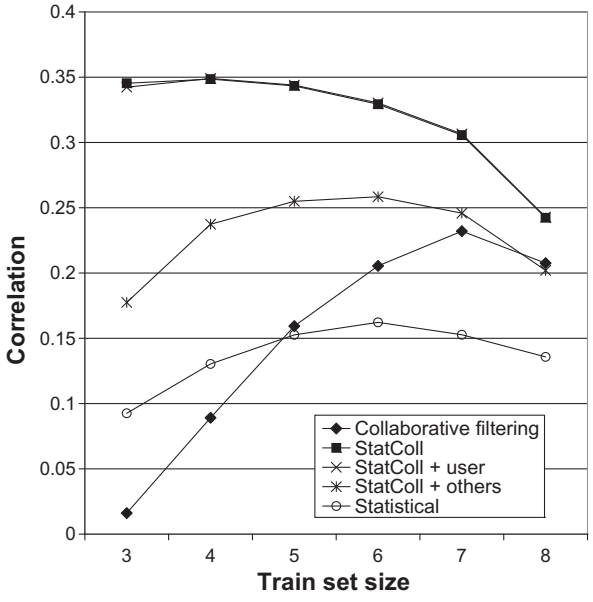

Fig. 9. Correlation for Sushi data.

improves. This hypothesis is supported by RMSE, which shows the same behaviour – Collaborative filtering improves after training set size = 20.

Pearson correlation is unaffected by the number of unpredicted objects, the results are in Fig. 4. The results are similar to Tau coefficient, except that Collaborative filtering is now better than Statistical, but still is worse than StatColls.

Number of evaluated object is in Fig. 6. All methods that used Statistical evaluated all objects in the test set (Statistical, StatColl + user, StatColl + others). Collaborative filtering has most unpredicted objects ($p < 0.001$), plain StatColl has about the half ($p < 0.001$). The number of evaluated objects decreases, because the training set size increases and there is a fixed number of objects (200).

Where Collaborative filtering wins is the training time ($p < 0.001$), represented in Fig. 5. The time is given in milliseconds per one training. As all other methods have to build the content based user preference model and compute the similarity of all the models, they spend very much time on that. We can see that Collaborative filtering and Statistical performed the fastest, StatColls are much worse. We must note that StatColls had to train for every user for every run - on the opposite in the real world; the user models would be trained already without the need to compute them again and again. The spikes on the lines are due to the random variations in time measurement.

5.2. Sushi dataset

We have used Sushi dataset Kamishima (2003) for the following experiment. The results for RMSE are in Fig. 7. Statistical and Collaborative filtering are the worst, StatColl and StatColl + user are the best. In this experiment, StatColl and StatColl + user performed very similarly, because there were almost no unpredicted objects for StatColl. The user model is used for prediction when there is no rating of the object by other users.

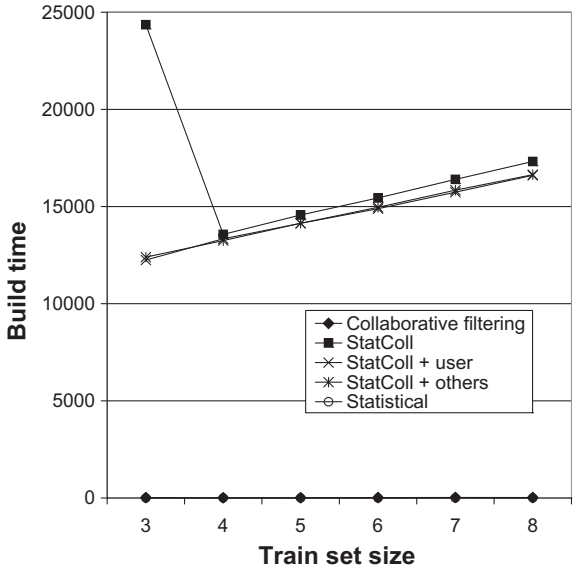

Fig. 10. Build time for Sushi dataset.

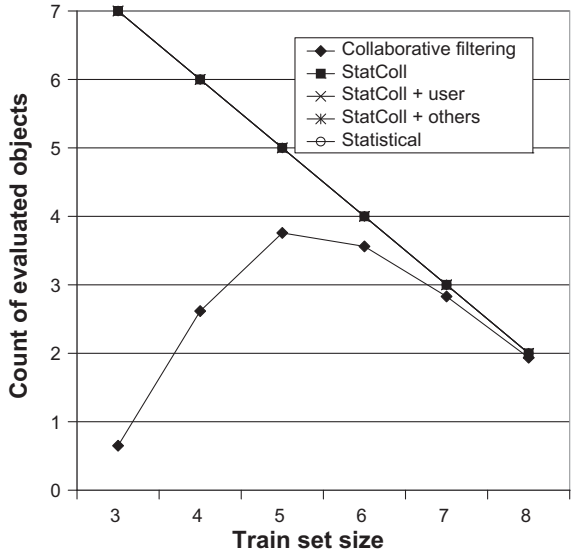

Fig. 11. Count of evaluated objects for Sushi dataset.

StatColl (and StatColl + user) is the clear winner for Tau coefficient (Fig. 8), other methods performed much worse. Similar results are also for Pearson correlation coefficient in Fig. 9.

The results for build time are in Fig. 10 and are the same as for Netflix dataset. The jump at training set size = 3 is due to the random variations in time measurement.

The number of evaluated objects however differs – Fig. 11 shows that only Collaborative filtering has few evaluated objects for small training sets and StatColl managed to evaluate all objects. This is thanks to the fact that StatColl can be estimate the similarity among all users, but Collaborative filtering can compare only users that rated at least one common object. When the training sets are this small (3 objects), the number of users that rated one of the three sushis is small to zero.

6. Conclusion

In this paper, we have proposed a way of computing the similarity of users using their preference models. This approach is suitable for cases with low number of users and low number of ratings. The proposed methods StatColls performed very well on standard datasets Netflix and Sushi against standard Collaborative filtering. The main advantage is the ability to estimate the similarity between all users, without the need to have at least some objects rated in common.

Acknowledgment

All the source codes can be downloaded at http://code.google.com/p/prefwork/ as SVN checkout or jar package.

This research was supported by projects SVV-2010-261312, GACR 201/09/H057, GACR P202/10/0761.

References

Agarwal, D., & Chen, B.-C. (2009). Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining KDD '09* (pp. 19–28). New York, NY, USA: ACM.

Basilico, J., & Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning ICML '04* (pp. 9). New York, NY, USA: ACM.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Gokhale, A., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the ACM SIGIR '99 workshop on recommender systems: Algorithms and evaluation*. Berkeley, California: ACM.

Clemen, R. T. (1996). *Making hard decisions: An introduction to decision analysis*. Belmont, California: Duxbury Press.

Eckhardt, A. (2007). Inductive models of user preferences for semantic web. In Pokorný, J., Snášel, V., & Richta, K. (Eds.), *DATESO 2007 volume 235 of CEUR Workshop Proceedings* (pp. 108–119). Matfyz Press, Praha.

Eckhardt, A., Pokorný, J., & Vojtáš, P. (2007). A system recommending top-k objects for multiple users preferences. In Martin, T. (Ed.), *2007 IEEE Conference on Fuzzy Systems IEEE Fuzzy systems* (pp. 1101–1106). London, United Kingdom.

Eckhardt, A., & Vojtáš, P. (2008). Considering data-mining techniques in user preference learning. In *2008 International Workshop on Web Information Retrieval Support Systems* (pp. 33–36).

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using Collaborative filtering to weave an information tapestry. *Communications of the ACM, 35*, 61–70.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (July 2001). Eigentaste: A constant time Collaborative filtering algorithm. Information Retrieval 4, 11.

Kamishima, T. (2003). Nantonac Collaborative filtering: Recommendation based on order responses. In *KDD '03:Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 583–588). New York, NY, USA: ACM.

Ko, S.-J., & Lee, J.-H. (2002). User preference mining through Collaborative filtering and content based filtering in recommender system. In *EC-WEB '02:Proceedings of the third international conference on e-commerce and web technologies* (pp. 244–253). London, UK: Springer-Verlag.

Lee, T. Q., Park, Y., & Park, Y.-T. (2007). A similarity measure for Collaborative filtering with implicit feedback. In *Proceedings of the 3rd international conference on intelligent computing: Advanced intelligent computing theories and applications. With aspects of artificial intelligence ICIC '07* (pp. 385–397). Berlin,Heidelberg: Springer-Verlag.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted Collaborative filtering for improved recommendations. In *Eighteenth national conference on artificial intelligence. American association for artificial intelligence* (pp. 187–192). CA, USA: Menlo Park.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, 13*, 393–408.

Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., McNee, S. M., Konstan, J. A., et al. (2002). Getting to know you: Learning new user preferences in recommender systems. In *IUI '02:Proceedings of the 7th international conference on intelligent user interfaces* (pp. 127–134). New York, NY, USA: ACM.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth. In *CHI '95:Proceedings of the SIGCHI conference on human factors in computing systems* (pp. 210–217). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Václav, B., Eckhardt, A., & Vojtáš, P. (2010). Prefshop – A web shop with user preference search capabilities. In *Proceedings of 2010 international workshop on web information retrieval support systems* (pp. 330–333). IEEE Computer Society.