Recommender System for Online Dating Service

Lukáš Brožovský[1] and Václav Petříček[1]

KSI MFF UK
Malostranské nám. 25,
Prague 1, Czech Republic
lbrozovsky@centrum.cz, petricek@acm.org

Abstract. Users of online dating sites are facing information overload that requires them to manually construct queries and browse huge amount of matching user profiles. This becomes even more problematic for multimedia profiles. Although matchmaking is frequently cited as a typical application for recommender systems, there is a surprising lack of work published in this area. In this paper we describe a recommender system we implemented and perform a quantitative comparison of two collaborative filtering (CF) and two global algorithms. Results show that collaborative filtering recommenders significantly outperform global algorithms that are currently used by dating sites. A blind experiment with real users also confirmed that users prefer CF based recommendations to global popularity recommendations. Recommender systems show a great potential for online dating where they could improve the value of the service to users and improve monetization of the service.

1 Introduction

Users of online dating sites are facing information overload. A typical dating site requires users to fill in lengthy questionnaires and matching is based on attributes selected. A typical search in the database either returns no match in the case when the query is too specific or, for general queries, returns a huge amount of profiles. Ranking of the results is difficult and usually requires users to assign weight to each of their requirement. Such approach is limited especially because different users express the same thing in different words. In the end users end up browsing large amount of irrelevant results. Query suggestion and query rewriting has been used in information retrieval to alleviate this problem but it becomes even more problematic for multimedia profiles.

Offline dating services have a very long history. With the advent of the web some of these companies started to put up web sites and build their online presence. Many new startups appeared but generally they mimicked the processes of offline dating agencies including questionnaires and matching algorithms.[1] In the last few years, a specific subgroup of online dating service web applications has emerged on the Internet. Their main idea is to let users store personal profiles and then browse and rate other users profiles. The registered user is

[1] http://www.date.com, http://www.match.com, http://www.perfectmatch.com, etc.

typically presented with random profiles (or with random profiles preselected with a simple condition - for example only men of certain age) and rates the profiles on a given numerical scale.[2] Online dating sites generally allow users to post and browse profiles for free and require payment for contact details. From the business model point of view the dating agencies should have high interest in improving the quality of recommendations they provide. Recommender systems [14] have been used in many areas. They have been popularized especially by recommenders at Amazon, Netflix, Movielens and others. Even though matchmaking is often cited as a potential area for recommender systems application there has been a surprising lack of work published in this area.

In this paper we describe a recommender system for online dating agency, benchmark algorithms on real world dataset and perform a study with real users to evaluate the quality of recommendations. Presented improvement in online dating and matchmaking has many benefits for users of the service as well as for the owners. These benefits include higher user satisfaction and loyalty, and also a better monetization of the service.

2 Related Work

Recommender systems [14] are a popular and successful way of tackling the information overload. Recommender systems have been popularized by applications such as Amazon [10] or Netflix recommenders[3]. The most widely used recommender systems are based on collaborative filtering algorithms. One of the first collaborative filtering systems was Tapestry [5]. Other notable CF systems include jester [6], Ringo [18], Movielens and Launch.com.

There are two classes of CF algorithms - memory based (e.g. user-user) and model based. Memory based systems require the whole rating matrix stored in memory. Typical example of a memory based algorithm is user-user k-nearest neighbor algorithm. Model based algorithms include methods based on rating matrix decomposition: for example SVD [15], MMMF [12], and other methods like item-item [16], Bayes-networks [1], personality diagnosis [11]. Collaborative filtering research has been addressing many areas including prediction accuracy, scalability, cold start problem, robustness and recommendation quality. Breese et al. [1] and [8] contain empirical comparisons of different types of algorithms in terms of prediction MAE and review of results in this area. Deshpande et al. [4] performed a comparison of topN item based recommender systems on several datasets including movies, credit card transactions, and others using metrics such as hit rate. In addition to these cross-validation experiments several studies with real users were performed. Ziegler et al [19] showed how topic diversification in the recommended set increases book recommendation quality as perceived by users. Recommendation explanation role has been recognized and/or used in [3, 17, 9]. Cosley et al. [3] compared several recommender systems in CiteSeer

[2] http://hotornot.com, http://libimseti.cz, http://chceteme.volny.cz
[3] http://amazon.com, http://netflix.com

digital library. Despite the amount of previous work none of it addressed dating in particular and none of the datasets used represented dating preferences.

Even though dating is often cited as a potential application for collaborative filtering based recommender systems and some dating sites claim to use proprietary collaborative techniques[4] there is a surprising lack of literature published on this problem.

3 Algorithms

Random Algorithm is a simple algorithm – every prediction is an uniformly distributed random value within the rating scale. For fixed user a and profile j it always predicts the same random rating $p_{a,j}$. This ensures that the predictions though random stay the same between runs.

Mean Algorithm is sometimes referred to as the Item Average Algorithm or the POP algorithm [1]. The prediction $p_{a,j}$ is calculated as the mean value of all the non-zero ratings for profile j ($\overline{r_{\cdot,j}}$). This algorithm is not personalized and completely ignores the information entered by the active user a. Random and Mean algorithms serve as our comparison baselines.

User-User Algorithm [7] is one of the best known collaborative filtering algorithms. When predicting ratings for active user a, the user database is first searched for users with similar ratings vectors to the user a – 'neighbors'. Ratings of the k most similar neighbors who rated item i are then used to calculate the prediction for the active user a and profile j:

$$p_{a,j} = \overline{r_a} + \xi \sum_{i=1}^{k} w(a, n_i)(r_{n_i,j} - \overline{r_{n_i}}) \tag{1}$$

where n_i is the i-th nearest neighbor, ξ is a normalizing factor, and $\overline{r_{n_i}}$ is the mean rating of the user n_i. As similarity we used the well known Pearson's correlation coefficient [13]:

$$w(a, j) = \frac{\sum_i (r_{a,i} - \overline{r_a})(r_{j,i} - \overline{r_j})}{\sqrt{\sum_i (r_{a,i} - \overline{r_a})^2 \sum_i (r_{j,i} - \overline{r_j})^2}} \tag{2}$$

where the summations over i are over the profiles which both users a and j have rated.

Our User-User algorithm implementation is parametrized by by two parameters i) $MinO$ – minimum number of common ratings between users necessary to calculate user-user similarity and ii) $MaxN$ – maximum number of user neighbors to be used during the computation. We refer to User-User algorithm with parameters $MinO = 5$, $MaxN = 50$ as "User-User (5,50)".

[4] http://www.match.com

Item-Item algorithm [16] uses a different view on the ratings matrix. Instead of utilizing the similarity between rows of the matrix as User-User does, it computes similarities between profiles.[5] We used adjusted Pearson correlation as item-item similarity:

$$w_{adj}(j,l) = \frac{\sum_i (r_{i,j} - \overline{r_i})(r_{i,l} - \overline{r_i})}{\sqrt{\sum_i (r_{i,j} - \overline{r_i})^2 \sum_i (r_{i,l} - \overline{r_i})^2}} \tag{3}$$

where the summations over i are over the users who rated both profiles j and l. Each pair of common ratings of two profiles comes from a different user with The adjusted version of the Pearson correlation subtracts each user's mean – otherwise the similarity computation would suffer from the fact that different rating scale. When making prediction $p_{a,j}$ for the active user a, the ratings of the active user a for the k most similar neighbors to profile j are used:

$$p_{a,j} = \overline{r_{\bullet,j}} + \xi \sum_{i=1}^{k} \tilde{w}(j, n_i)(r_{a,n_i} - \overline{r_{\bullet,n_i}}) \tag{4}$$

where n_i is the i-th most similar profile to profile j that user a rated, ξ is a normalizing factor, and $\overline{r_{\bullet,u}}$ is the mean rating of profile u.

Similar to User-User algorithm, our implementation of Item-Item algorithm is also parametrized by two parameters i) $MinO$ – minimum number of common ratings between profiles necessary to calculate item-item similarity and ii) $MaxN$ – maximum number of item neighbors to be used during the computation. We refer to the Item-Item algorithm with parameters $MinO = 5$, $MaxN = 50$ simply as 'Item-Item (5,50)'.

4 Libimseti Dataset

The dataset we used consists of data from a real online dating service – Libimseti. A snapshot of the rating matrix from July 2005[6] was used in this study. In the dataset users appear in two different roles: i) as active providers of ratings for photographs of others – we will denote them as 'users' in this situation and ii) as objects on photographs when they are rated by others – we will refer to them in this situation as 'profiles'.

Overall the dataset contains 194,439 users, who provided 11,767,448 ratings. The sparsity of the matrix is 0.03%. Table 1 compares our dataset to the Movielens [7] and Jester [6] datasets that are well known in collaborative filtering research community. We can see that Libimseti is much larger but also sparser than the other two datasets. The distribution of number of ratings provided by a single user and distribution of the rating value are shown in Figure 1. The similarity distributions for ratings matrix rows and columns are shown in Figure 2.

[5] Please note that in the dating service environment, both users and items could be seen as representing users.

[6] http://www.ksi.ms.mff.cuni.cz/~petricek/data/

Table 1. Data sets overview. An overview of the data set's basic characteristics. An *active user* is user who have rated another profile at least once. The last three rows contain statistics of ratings normalized to $0-1$ scale.

	MovieLens	**Jester**	**LibimSeTi**
total users	9,746	73,521	194,439
users with ratings	6,040	73,421	128,394
items with ratings	3,706	100	120,393
ratings	**1,000,209**	**4,136,360**	**11,767,448**
density	10.5302 ‰	0.7652 ‰	0.3113 ‰
max ratings from 1 user	2,314	100	15,060
max ratings for 1 profile	3,428	73,413	69,365
rating (mean/med/sd)	0.64/0.75/0.27	0.54/0.57/0.26	0.50/0.44/0.36

Fig. 1. Distribution of number of ratings. On the left is the distribution of number of ratings provided by single user. On the right then the distribution of the rating value.

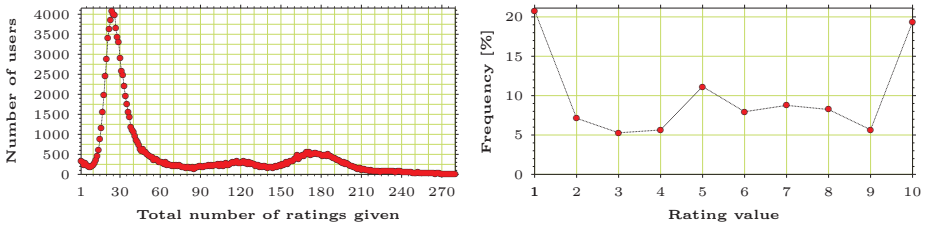
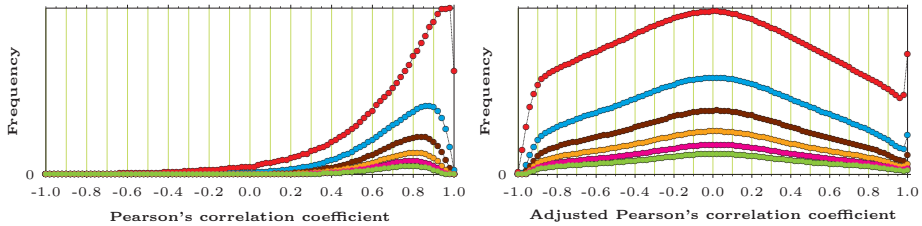


Fig. 2. Similarities distributions. User-User similarity and Item-Item similarity distribution. Each graph displays 6 distributions for different minimum overlap parameter (from top to bottom $MinO$ = 5,10,15,20,25,30).



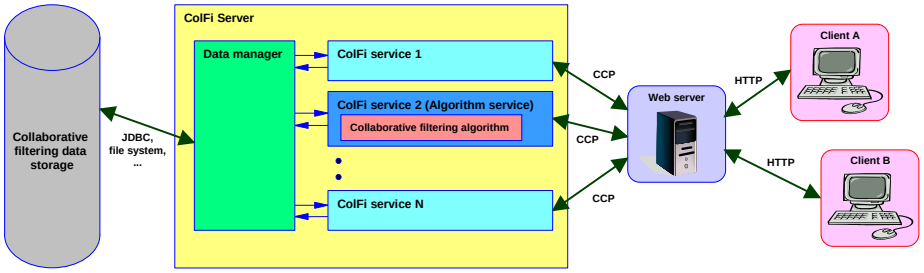
5 ColFi Recommender System

We implemented a domain independent and freely available[7] recommender system that has been tested on four different datasets [2]. ColFi System architecture has been designed to be flexible yet simple enough so that developers can focus on collaborative filtering algorithms. Core of the system is a stateless TCP/IP

[7] http://colfi.wz.cz

server. It consists of a global data manager, a set of ColFi services and a communication module implementing CCP - ColFi Communication Protocol. The data manager serves as the main data provider for all the components in the system and ColFi services expose API to clients). Particular data manager and ColFi services implementations are provided as server plug-ins. Figure 3 summarizes ColFi architecture design.

Fig. 3. ColFi architecture. The communication protocol between the data storage and the data manager is implementation dependent.



System was implemented as a stateless TCP/IP client-server solution to achieve simplicity, platform independence, scalability and good performance. Each ColFi service instance has its own unique TCP/IP port number and is independent of all the other ColFi services. Multiple client connections are therefore supported and handled in parallel. We implemented the system in Java. The implementation allows easy addition of collaborative filtering algorithms. We use a lazy approach to similarity computations and store the similarities in a local cache object. MySQL server is used as a back end but the database is fully abstracted in the code. For more details on the implementation see [2].

ColFi Communication Protocol is similar to popular Java RMI. CCP is a very simple protocol and it supports only invocation of limited set of remote methods. That makes it very fast unlike SOAP Webservices) and does not require special setup on the client side unlike Java RMI, EJB, JSP, or Servlets.

Data Manager is the heart of the ColFi system. It is a general interface that provides all the other ColFi components with data necessary for collaborative filtering. There is always exactly one data manager instance per server instance in a form of a plug-in for the ColFi server and the actual behavior can differ significantly for different implementations. The data manager can for example cache data in the memory or it can directly access database or special file system. In some other cases, read only implementation may be desired. Although both cached and direct access is implemented, only cached variants achieve reasonable performance. Direct access is generally usable only with small data sets.

ColFi Service is the only part of the system exposed to the outside world. Each service has its own TCP/IP server port number where it listens for incoming client requests. ColFi services are stateless, so each request is both atomic

and isolated and should not affect any other requests. This allows simple parallel handling of multiple client connections on the server. The ColFi service is not limited to do collaborative filtering, it can also provide statistics for example.

6 Benchmarks

We compare the implemented algorithms on Libimseti dataset using three types of cross-validation. In each of the scenarios we provide algorithms with one subset of ratings for training and withhold other subset of ratings for testing the prediction accuracy.

6.1 Setup

We used three types of cross-validation: i) AllButOne ii) GivenRandomN and iii) production. Each validation uses NMAE as a metric of prediction quality:

$$\mathrm{NMAE} = \frac{\frac{1}{c}\sum_{k=1}^{c} |\tilde{p}_{ij}^{k} - r_{ij}|}{r_{max} - r_{min}} \tag{5}$$

It is possible that some algorithms fail to predict certain ratings due to lack of data. These cases are ignored and not included in NMAE computation[8] and only the total number of such unsuccessful predictions is reported.

AllButOne Validation is the most commonly used benchmark among many collaborative filtering papers [6, 1]. It is the simplest protocol of the three. The idea behind this protocol is to select exactly one rating and hide it from the tested algorithm. Algorithm is then asked to predict that one hidden rating and the prediction error is the absolute difference between the hidden rating value and the predicted rating value. We reapeat this for *every* single rating in the test data set.

GivenRandomX Validation is intended to examine the algorithm's performance with less data available from the active user. This situation is sometimes referred to as user cold-start and happens when a user is new to the system and did not provide enough ratings yet.

The validation works on subset of users who provided more than 100 ratings. We split the ratings into two: a training group T containing 99 random ratings from each user and a test set with remaining ratings. Group T is then used to generate 99 training sets t_i for $i = 1..99$ in such a way that $\bar{t}_i{=}i$ and $\forall_i t_i \subset t_{i+1}$. These sets represent the ratings gradually provided by active user. The result of the GivenRandomX validation is a graph of the NMAE values as a function of training set size i.

[8] Our experiments showed that this does not affect the ordering of algorithms.

Production Validation is designated to test the algorithm as a part of the ColFi System. The entire test data set is split into two disjunctive sets: the initial set and the simulation set. Both sets are about the same size. ColFi server is started with an algorithm being tested. Exactly N_{max} clients (users) insert ratings from the simulation set of the test data set into the initial set through the ColFi server interface (this includes additions of new users to the systems as well). Ratings from the simulation part are selected at random. Prior to every single rating insertion on the server side, a prediction for the rating being inserted is computed. After every K-sized block of insertions, the NMAE of that set/block is calculated. The Production validation result is a graph of NMAE values based on the index of the K-sized block of insertions.

6.2 Results

Benchmarks were run on PC, 2x AMD Opteron™ 250 (2.4GHz, 1MB), 8 GB RAM, running Red Hat Linux 3.2.3-47. The variable parameters of the protocol were set as follows: K = 10,000 and N_{max} = 100 (hundred concurrent clients).

AllButOne Validation Results are summarized in Table 2. First, an expected observation, that bigger neighborhood results in a better accuracy of the User-User algorithm. Further experiments did confirm that more than 50 neighbors do not significantly improve the performance. Item-Item algorithm with neighborhood size of 50 was used for the benchmarks. Interesting observation is the small difference in NMAE values for Mean and CF algorithms. Overall User-User algorithm has best performance. Mean algorithm performs surprisingly well due to strong global component in user preferences.

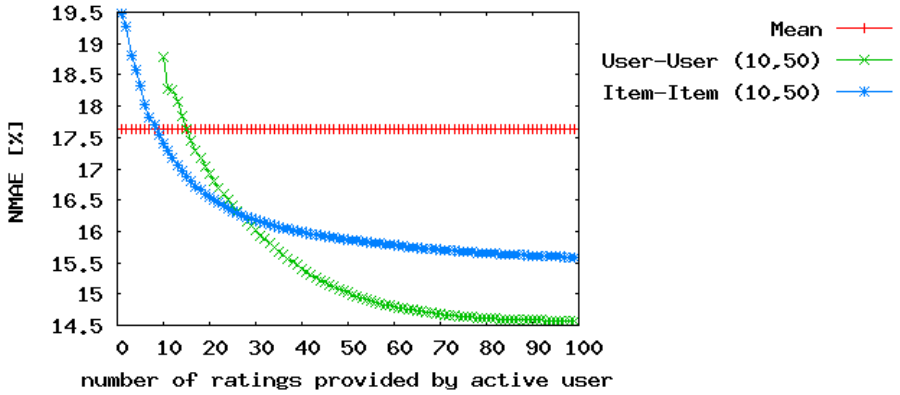
Table 2. AllButOne results. Columns contain: i) NMAE of the algorithm prediction calculated in the data set's original scale ii) Number of all skipped rating predictions (due to insufficient data)

algorithm	**NMAE**	**skipped**
Random	39.72%	0
Mean	15.69%	24,785
User-User (5,10)	14.54%	74,560
User-User (5,50)	13.85%	74,560
User-User (10,10)	13.56%	174,352
User-User (10,50)	13.34%	174,352
Item-Item (10,50)	14.06%	252,623

GivenRandomX Validation Results are presented as NMAE graph in the Figure 4. For readability the Random algorithm results are not displayed.

Algorithms show improved performance in terms of NMAE values for the first few ratings inserted. After the user enters about 30-40 ratings, algorithm's accuracies get stabilize (algorithm has enough information to derive the active user type). Overall, the User-User algorithm has generally lowest NMAE values.

Fig. 4. GivenRandomX benchmark results. NMAE values based on the number of ratings given by a potential new user. The graph shows the behavior of the algorithms as they get more information about particular user. Data for the Random algorithm are not displayed as it has relatively high constant NMAE value of 38.27%.



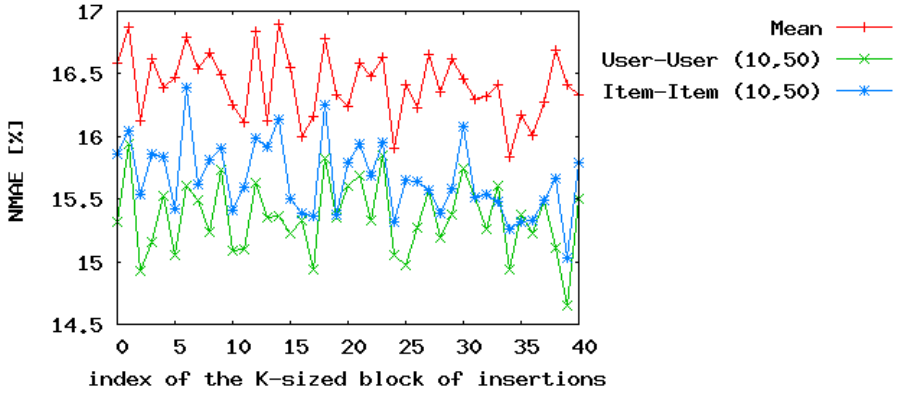
Production Validation Results are presented in the Figure 5. Results for the worst performing Random algorithm are again not displayed in the NMAE graphs to improve the legibility of the figure. The performance ordering of algorithms corresponds to the AllButOne and GivenRandomN results. During the course of this experiment we also verified that ColFi recommender can handle significant load at the level of real world application traffic.

7 User experiment

We conducted a web based experiment with human subjects. Each participant was asked to provide ratings for 150 random profiles. Based on the ratings provided we generated two recommendation lists of top 10 profiles. We tested the Random, Mean, and User-User algorithms. User-User (10,50)[9] was selected as the best performing collaborative algorithm, Mean represented the currently deployed functionality at Libimseti.cz and Random served as a baseline. Recommendation lists contained only profiles that the active user did not rate yet. Each

[9] MinO=10, MaxN=50

Fig. 5. Production benchmark results. NMAE values based on the index of K-sized set/block of insertions. Data for the Random algorithm are not displayed as it has relatively high mean NMAE value of 39.12%.



of the recommendation lists was generated using one of the ColFi algorithms. The two lists were then presented to the user who was asked to selected the list that he perceives as better recommendation. The order of the recommendation lists was randomized to avoid bias. Gender information was used to provide opposite sex recommendations.

Participants of the study were 111 users recruited mainly through the collab mailing list.[10] From July to August 2006 we collected 14,057 ratings for 1,000 profiles (matrix sparsity 11.39‰). Table 3 summarizes the outcomes of individual "recommendation duels" between all pairs of algorithms tested. User-User algorithm was perceived by users as best although global mean algorithm has won a surprising fraction of duels.

Table 3. User study results – outcomes of duels between the three algorithms tested. Each cell contains the percentage of duels where algorithm on the left won over algorithm on top. Cells where algorithm on left was percieved better more than 50% of time are in bold.

	Random	**Mean**	**User-User**
Random	-	16.87%	12.50%
Mean	**83.13%**	-	35.62%
User-User	**87.50%**	**64.38%**	-

The Random algorithm "lost" against both collaborative filtering algorithm and the Mean algorithm. On the other hand, the fractions won in the table for

[10] http://www2.sims.berkeley.edu/resources/collab/

the Random algorithm – 12.5% and 16.87% – are quite high, which indicates that percieved profile quality is clustered due to self-selection bias (good looking people are more likely to put their photo online). The surprising success of the Mean algorithm – 35.62% wins against collaborative filtering algorithm – suggests a strong universal preference that is shared by users – in fact this is a prerequisitive for any beauty contest to be possible.

8 Discussion

We have shown that collaborative filtering based recommender systems can provide good recommendations to users of online dating services. We demonstrated this using cross-validation on a snapshot of a real world dating service. User-User and Item-Item algorithm outperformed global popularity in terms of prediction NMAE 3.08% and 2.04% respectively. We also verified that this difference in recommendation quality is noticable to users who preferred recommendations by User-User algorithm to global popularity recommendations.

A logical continuation of our work is a more complete evaluation of current state of the art recommender systems including model based collaborative filtering methods in the setting of dating services. These methods could also help to address scalability and performance issues. Domain specific improvements may be possible.

User interface may introduce bias in the sense that users instead of providing their personal preference try to guess the global preference. This reduces the usefulness of ratings provided. It remains an open issue how to best design an interface which motivates users to provide sufficient amount of truthful ratings. Application of index structures to speed up nearest neighbor search is an interesting research direction.

Recommendations can be further improved by hybrid algorithms. These algorithms are combining the collaborative filtering approach with content information. Another problem specific to dating is that "A likes B" does not imply "B likes A". Therefore each user should be probably presented with recommendations of such users, who are also interested in him/her. There is a need for reciprocal matching algorithms.

9 Acknowledgments

We would like to thank our three anonymous reviewers for their ideas and suggestions. We would also like to thank Oldřich Neuberger for providing the anonymized Libimseti dataset and to Tomáš Skopal for his valuable comments.

References

1. John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. pages 43–52, 1998.

2. Lukáš Brožovský. Recommender system for a dating service. Master's thesis, KSI, MFF UK, Prague, Czech Republic, 2006.
3. Dan Cosley, Steve Lawrence, and David M. Pennock. REFEREE: An open framework for practical testing of recommender systems using researchindex. In *28th International Conference on Very Large Databases, VLDB 2002*, 2002.
4. Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.
5. David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.
6. Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
7. Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM Press, 1999.
8. Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.
9. H Kautz, B Selman, and M Shah. Referral web: combining social networks and collaborative filtering. *Communications of the ACM*, 1997.
10. G Linden, B Smith, and J York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 2003.
11. David Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, UAI 2000*, pages 473–480, 2000.
12. JDM Rennie and N Srebro. Fast maximum margin matrix factorization for collaborative prediction. *Proceedings of the 22nd International Conference on Machine Learning*, 2005.
13. P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186. ACM, 1994.
14. Paul Resnick and Hal R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.
15. B Sarwar, G Karypis, J Konstan, and J Riedl. Incremental singular value decomposition algorithms for highly scalable recommender systems. *Fifth International Conference on Computer and Information Science*, 2002.
16. Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the tenth international conference on World Wide Web*, pages 285–295, 2001.
17. Rashmi R. Sinha and Kirsten Swearingen. Comparing recommendations made by online systems and friends. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries*, 2001.
18. S. Upendra. Social information filtering for music recommendation, 1994.
19. Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM Press, 2005.