Data quality issues in collaborative filtering

Miha Grčar, Dunja Mladenič, and Marko Grobelnik

J.Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia
Miha.Grcar@ijs.si,
WWW home page: http://kt.ijs.si/

Abstract. In this paper, we present our experience in applying collaborative filtering to real-life corporate data. The quality of collaborative filtering recommendations is highly dependent on the quality of the data used to identify users' preferences. To understand the influence that highly sparse server-side collected data has on the accuracy of collaborative filtering, we ran a series of experiments in which we used publicly available datasets and, on the other hand, a real-life corporate dataset that does not fit the profile of ideal data for collaborative filtering. We have performed a series of experiments on two standard data sets (EachMovie and Jester) and a real-life corporate data.

1 Introduction and motivation

The goal of collaborative filtering is to explore a vast collection of items in order to detect those which might be of interest to the active user. In contrast to content-based recommender systems which focus on finding contents that best match the user's query, collaborative filtering is based on the assumption that similar users have similar preferences. It explores the database of users' preferences and searches for users that are similar to the active user. The active user's preferences are then inferred from preferences of the similar users. One of the main advantages of pure collaborative filtering is that it ignores the form and the content of items and can therefore also be applied to non-textual items.

The accuracy of collaborative filtering recommendations is highly dependant on the quality of the users' preferences database. In this paper we would like to emphasize the differences between applying collaborative filtering to publicly available datasets and, on the other hand, to a dataset derived from real-life corporate Web logs. The latter does not fit the profile of ideal data for collaborative filtering.

The rest of this paper is arranged as follows. In Sections 2 and 3 we discuss collaborative filtering algorithms and data quality for collaborative filtering. Our evaluation platform and the three datasets used in our experiments are described in Sections 4 and 5. In Sections 6 and 7 the experimental setting and the evaluation results are presented. The paper concludes with the discussion and some ideas for future work (Section 8).

2 Collaborative filtering

There are basically two approaches to the implementation of a collaborative filtering algorithm. The first one is the so called "lazy learning" approach (also known as the memory-based approach) which skips the learning phase. Each time it is about to make a recommendation, it simply explores the database of user-item interactions. The model-based approach, on the other hand, first builds a model out of the user-item interaction database and then uses this model to make recommendations. "Making recommendations" is equivalent to predicting the user's preferences for unobserved items.

The data in the user-item interaction database can be collected either explicitly (explicit ratings) or implicitly (implicit preferences). In the first case the user's participation is required. The user is asked to explicitly submit his/her rating for the given item. In contrast to this, implicit preferences are inferred from the user's actions in the context of an item (that is why the term "user-item interaction" is used instead of the word "rating" when referring to users' preferences in this paper). Data can be collected implicitly either on the client side or on the server side. In the first case the user is bound to use modified client-side software that logs his/her actions. Since we do not want to enforce modified client-side software, this possibility is usually omitted. In the second case the logging is done by a server. In the context of the Web, implicit preferences can be determined from access logs that are automatically maintained by Web servers.

Collected data is first preprocessed and arranged into a user-item matrix. Rows represent users and columns represent items. Each matrix element is in general a set of actions that a specific user took in the context of a specific item. In most cases a matrix element is a single number representing either an explicit rating or a rating that was inferred from the user's actions.

Since a user usually does not access every item in the repository, the vector (i.e. the matrix row), representing the user, is missing some/many values. To emphasize this, we use the terms "sparse vector" and "sparse matrix".

The most intuitive and widely used algorithm for collaborative filtering is the so called k-Nearest Neighbors algorithm which is a memory-based approach. Technical details can be found, for example, in Grcar (2004). The algorithm is as follows:

1. Represent each user by a sparse vector of his/her ratings.
2. Define the similarity measure between two sparse vectors. In this paper, we consider two widely used measures: (i) the Pearson correlation coefficient which is used in statistics to measure the degree of correlation between two variables (Resnick et al. (1994)), and (ii) the Cosine similarity measure which is originally used in information retrieval to compare between two documents (introduced by Salton and McGill in 1983).
3. Find k users that have rated the item in question and are most similar to the active user (i.e. the user's neighborhood).

4. Predict the active user's rating for the item in question by calculating the weighted average of the ratings given to that item by other users from the neighborhood.

3 Sparsity problem and data quality for collaborative filtering

The fact that we are dealing with a sparse matrix can result in the most concerning problem of collaborative filtering – the so called sparsity problem. In order to be able to compare two sparse vectors, similarity measures require some values to overlap. What is more, the lower the amount of overlapping values, the lower the relialibility of these measures. If we are dealing with high level of sparsity, we are unable to form reliable neighborhoods. Furthermore, in highly sparse data there might be many unrated (unseen) items and many inactive users. Those items/users, unfortunately, cannot participate in the collaborative filtering process.

Sparsity is not the only reason for the inaccuracy of recommendations provided by collaborative filtering. If we are dealing with implicit preferences, the ratings are usually inferred from the user-item interactions, as already mentioned earlier in the text. Mapping implicit preferences into explicit ratings is a non-trivial task and can result in false mappings. The latter is even more true for server-side collected data in the context of the Web since Web logs contain very limited information. To determine how much time a user was reading a document, we need to compute the difference in time-stamps of two consecutive requests from that user. This, however, does not tell us weather the user was actually reading the document or he/she, for example, went out to lunch, leaving the browser opened. What is more, the user may be accessing cached information (either from a local cache or from an intermediate proxy server cache) and there is no way to detect these events on the server side.

Also, if a user is not logged in and he/she does not accept cookies, we are unable to track him/her. In such case, the only available information that could potentially help us to track the user is his/her IP address. However, many users can share the same IP and, what is more, one user can have many IP addresses even in the same session. The only reliable tracking mechanisms are cookies and requiring users to log in in order to access relevant contents.

From this brief description of data problems we can conclude that for applying collaborative filtering, explicitly given data with low sparsity are preferred to implicitly collected data with high sparsity. The worst case scenario is having highly sparse data derived from Web logs. When so, why would we want to apply collaborative filtering to Web logs? The answer is that collecting data in such manner requires no effort from the users and also, the users are not obliged to use any kind of specialized Web browsing software. This "conflict of interests" is illustrated in Figure 1.

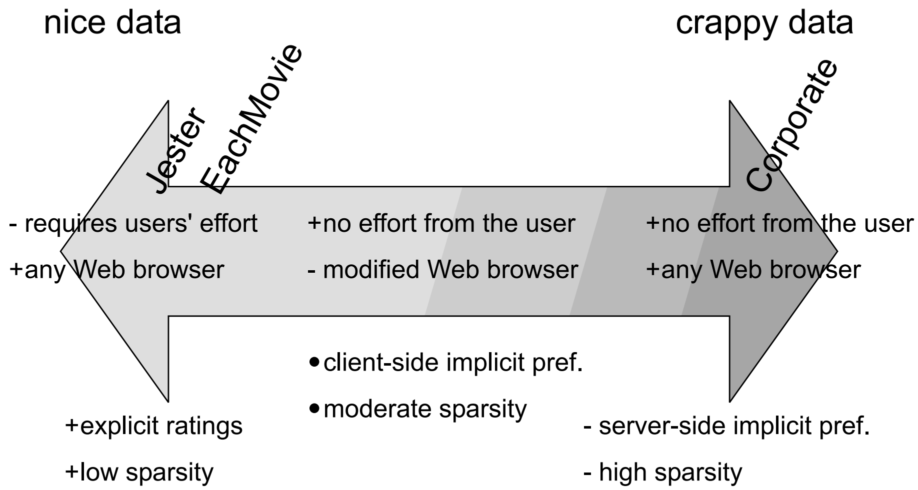

Fig. 1. Data characteristics that influence the data quality, and the positioning of the three datasets used in our experiments, according to their properties.

4 Evaluation platform

To understand the influence that highly sparse server-side collected data has on the accuracy of collaborative filtering, we built an evaluation platform. This platform is a set of modules arranged into a pipeline. The pipeline consists of the following four consecutive steps: (i) importing a user-item matrix (in the case of implicit preferences, data needs to be preprocessed prior to entering the pipeline), (ii) splitting data into a training set and a test set, (iii) setting a collaborative filtering algorithm (in the case of the kNN algorithm we also need to specify a similarity measure) and an evaluation protocol, (iv) making predictions about users' ratings and collecting evaluation results. The platform is illustrated in Figure 2.

Let us briefly discuss some of these stages. In the process of splitting the data into a training set and a test set, we randomly select a certain percentage of users (i.e. rows from the user-item matrix) that serve as our training set. The training set is, in the case of the kNN algorithm, used to search for neighbors or, in the case of model-based approaches, as a source for building a model. Ratings from each user from the test set are further partitioned into “given” and “hidden” ratings, according to the evaluation protocol. For example, 30% of randomly selected ratings from a particular user are hidden, the rest are treated as our sole knowledge about the user (i.e. given ratings). Given ratings are used to find neighbors in the training set, while hidden ratings are used to evaluate the accuracy of the selected collaborative filtering algorithm. The algorithm predicts the hidden ratings and since we know their actual values, we can compute the mean absolute error (MAE) or apply some other evaluation metric.

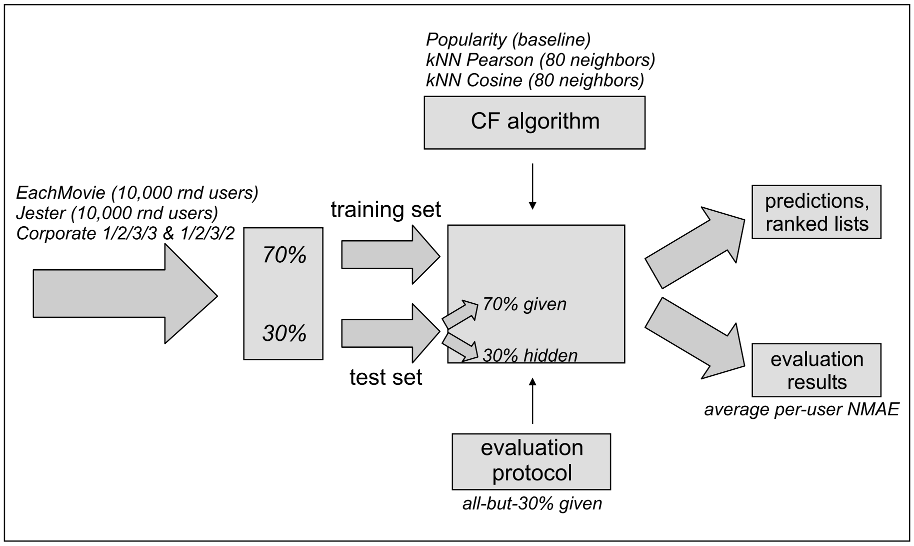

Fig. 2. The evaluation platform. The notes in *italics* illustrate our experimental setting (see Section 6).

5 Data description

For our experiments we used three distinct datasets. The first dataset was EachMovie (provided by Digital Equipment Corporation) which contains explicit ratings for movies. The service was available for 18 months. The second dataset with explicit ratings was Jester (provided by Goldberg et al.) which contains ratings for jokes, collected over a 4-year period. Users were using a scrollbar to express their ratings – they had no notion of actual values. The third dataset was derived from real-life corporate Web logs. The logs contain accesses to an internal digital library of a fairly large company. The time-span of acquired Web logs is 920 days. In this third case the users' preferences are implicit and collected on the server side, which implies the worst data quality for collaborative filtering.

In contrast to EachMovie and Jester, Web logs first needed to be extensively preprocessed. Raw logs contained over 9.3 million requests. First, failed requests, redirections, posts, and requests by anonymous users were removed. We were left with slightly over 1.2 million requests (14% of all the requests). These requests, however, still contained images, non-content pages (such as index pages), and other irrelevant pages. What is more, there were several different collections of documents in the corporate digital library. It turned out that only one of the collections was relevant for the application of collaborative filtering. Thus, the amount of potentially relevant requests dropped drastically. At the end we were left with only slightly over 20,500 useful requests, which is 0.22% of the initial database size.

The next problem emerged from the fact that we needed to map implicit preferences contained in log files, into explicit ratings. As already explained, this is not a trivial task. The easiest way to do this is to label items as 1 (accessed) or 0 (not accessed) as also discussed in Breese et al. (1998). The downside of this kind of mapping is that it does not give any notion of likes and dislikes. Claypool et al. (2001) have shown linear correlations between the time spent reading a document and the explicit rating given to that same document by the same user (this was already published by Konstan et al. (1997)). However, their test-users were using specialized client-side software, which made the collected data more reliable (hence, in their case, we talk about client-side implicit preferences). Despite this fact we decided to take reading times into account when preprocessing Web logs.

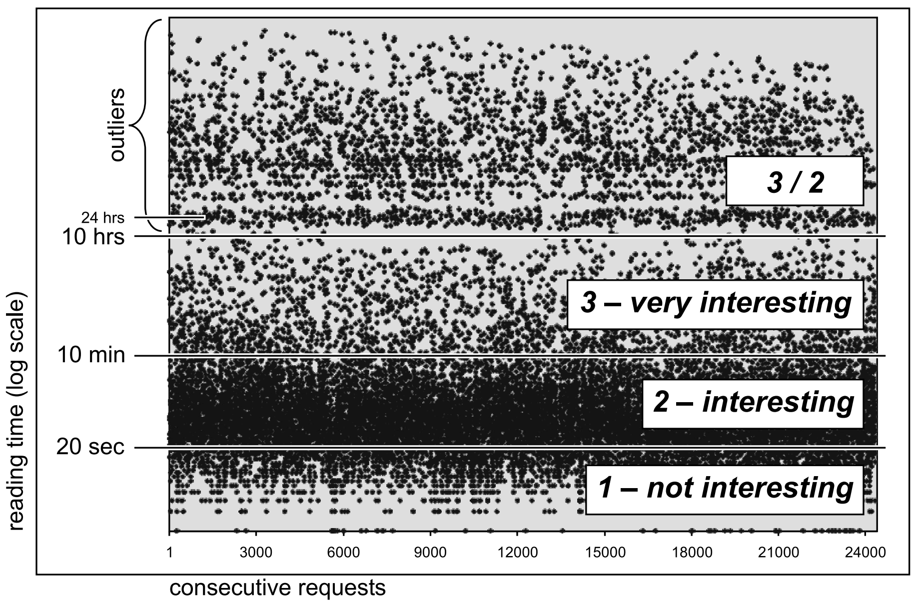

Fig. 3. Mapping implicit preferences contained in the corporate Web logs onto a discrete 3-score scale.

We plotted reading times inferred from consecutive requests onto a scatter plot shown in Figure 3. The X-axis shows requests ordered by their time-stamps, and the y-axis shows the inferred reading time on a logarithmic scale. We can see that the area around 24 hours is very dense. These are the last accesses of a day. People went home and logged in again the next day, which resulted in approximately 24-hour "reading" time. Below the 24-hour line, at approximately 10-hour reading time, a gap is evident. We decided to use this gap to define outliers – accesses above the gap are clearly outliers. We decided to map reding times onto a discrete 3-score scale (scores being 1="not interesting", 2="interesting", and 3="very interesting"). Somewhat ad-hoc (intuitively) we defined two more boundaries: one at 20 seconds and another at 10 minutes. Since items were research papers and 20 seconds is merely enough to browse through the abstract, we decided to label documents with reading times below 20 seconds as "not interesting". Documents with reading times between 20 seconds and 10 minutes were labelled as "interesting" and documents with reading times from 10 minutes to 10 hours were labelled as "very interesting". We decided to keep the outliers due to the lack of data. In the first scenario they were labelled as "very interesting" and in the second one as "interesting". Since we had no reliable knowledge about the outliers, the second scenario should have minimized the error we made by taking them into account.

Table 1 shows the comparison between the three datasets. It is evident that a low number of requests and somewhat ad-hoc mapping onto a discrete scale are not the biggest issues with our corporate dataset. The concerning fact is that the average number of ratings per item is only 1.22, which indicates extremely poor overlapping. Sparsity is consequently very high, 99.93%. The other two datasets are much more promising. The most appropriate is the Jester dataset with very low sparsity, followed by EachMovie with higher sparsity but still relatively high average number of ratings per item. Also, the latter two contain explicit ratings, which means that they are more reliable than the corporate dataset (see also Figure 1).

	Ratings		Size			Sparsity		
	Explicit/ implicit	Scale	Num of users	Num of items	Num of ratings	%**	Avg # of r'tings/usr	Avg # of ratings/item
EachMovie	Explicit	Discrete 0–5	61,131	1,622	2,558,871	97.42	41.86	1,577.60
Jester	Explicit	Continuous –10 – +10	73,421	100	4,136,360	43.66	56.34	41,363.60
Corporate	Implicit	Discrete 1–3*	1,850	16,941	20,669	99.93	11.17	1.22

*after preprocessing
**computed as the number of missing values divided by the user-item matrix size (i.e. the number of rows times the number of columns)

Table 1. The comparison between the three datasets.

6 Experimental setting

We ran a series of experiments to see how the accuracy of collaborative filtering recommendations differs between the three datasets (from EachMovie and Jester we considered only 10,000 randomly selected users to speed up the evaluation process). First, we randomly selected 70% of the users as our training set (the remaining 30% were our test set). Ratings from each user in the test set were further partitioned into "given" and "hidden" ratings according to the "all-but-30%" evaluation protocol. The name of the protocol implies that 30% of all the ratings were hidden and the remaining 70% were used to form neighborhoods in the training set.

We applied three variants of memory-based collaborative filtering algorithms: (i) k-Nearest Neighbors using the Pearson correlation (kNN Pearson), (ii) k-Nearest Neighbors using the Cosine similarity measure (kNN Cosine), and (iii) the popularity predictor (Popularity). The latter predicts the user's ratings by simply averaging all the available ratings for the given item. It does not form neighborhoods and it provides each user with the same recommendations. It serves merely as a baseline when evaluating collaborative filtering algorithms (termed "POP" in Breese et al. (1998)). For kNN variants, we used a neighborhood of 80 users (i.e. k=80), as suggested in Goldberg et al. (2001). We decided to evaluate both variants of the corporate dataset (the one where the outliers were labelled as "very interesting", referred to as "1/2/3/3", and the one where the outliers were labelled as "interesting", referred to as "1/2/3/2").

For each dataset-algorithm pair we ran 5 experiments, each time with a different random seed (we also selected a different set of 10,000 users from EachMovie and Jester each time). When applying collaborative filtering to the variants of the corporate dataset, we made 10 repetitions (instead of 5) since these datasets were smaller and highly sparse, which resulted in less reliable evaluation results. Thus, we ran 90 experiments altogether.

We decided to use normalized mean absolute error (NMAE) as the accuracy evaluation metric. We first computed NMAE for each user and then we averaged it over all the users (termed "per-user NMAE") (see Herlocker et al. (2004)). MAE is extensively used for evaluating collaborative filtering accuracy and was normalized in our experiments to enable us to compare evaluation results from different datasets.

7 Evaluation results

Our evaluation results are shown in Figure 4. The difference between applying kNN Pearson and kNN Cosine to EachMovie is statistically insignificant (we used two-tailed paired Student's t-Test to determine if the differences in results are statistically significant). However, they both significantly outperform Popularity. In the case of Jester, which has the smallest degree of sparsity, kNN Pearson slightly, yet significantly outperforms kNN Cosine. Again, they both significantly outperform Popularity. Evaluation results from the corporate datasets (two variants of the same dataset, more accurately) show that predictions are less accurate and that NMAE value is relatively unstable (hence the large error bars showing standard deviations of NMAE values). The main reason for this is low/no overlapping between values (i.e. extremely high sparsity), which results in inability to make several predictions. In the first scenario (i.e. with the 1/2/3/3 dataset) we can see that the differences in NMAE of kNN Pearson, kNN Cosine and Popularity are all statistically insignificant. In the second sce-

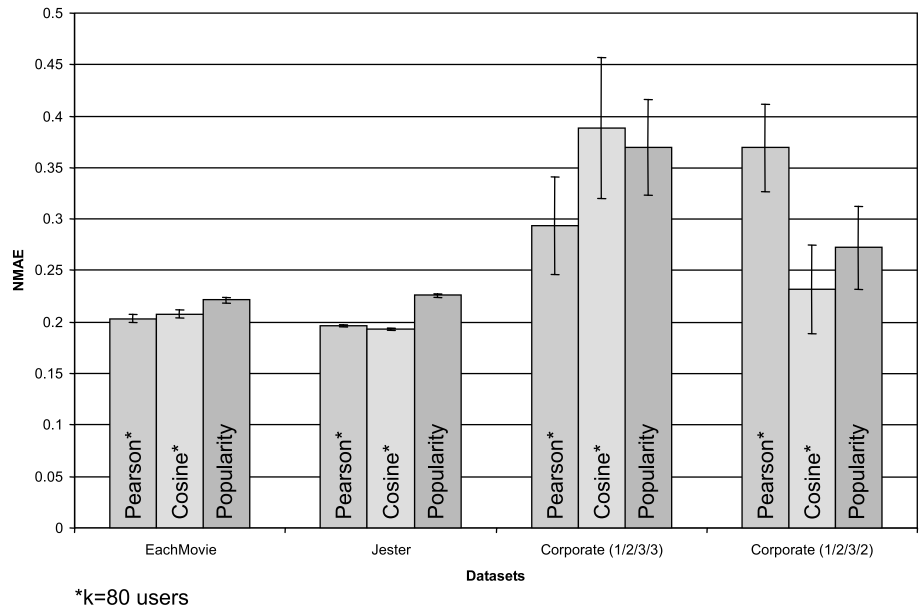

Fig. 4. The evaluation results.

nario (i.e. with the 1/2/3/2 dataset), however, kNN Pearson outperforms kNN Cosine and Popularity, while the accuracies of kNN Cosine and Popularity are not significantly different.

8 Discussion and future work

What is evident from the evaluation results is that the corporate dataset does not contain many overlapping values and that this represents our biggest problem. Before we will really be able to evaluate collaborative filtering algorithms on the given corporate dataset, we will need to reduce its sparsity. One idea is to apply LSI (latent semantic indexing) (Deerwester et al. (1990)) or to use pLSI (probabilistic latent semantic indexing) (Hofmann (1999)) to reduce the dimensionality of the user-item matrix, which consequently reduces sparsity. Another idea, which we believe is even more promising in our context, is to incorporate textual contents of the items. There were already some researches done on how to use textual contents to reduce sparsity and improve the accuracy of collaborative filtering (Melville et al. (2002)). Luckily we are able to obtain textual contents for the given corporate dataset.

What is also evident is that mapping implicit into explicit ratings has great influence on the evaluation results. We can see that going from Corporate 1/2/3/3 to Corporate 1/2/3/2 is fatal for kNN Pearson (in contrast to kNN Cosine). This needs to be investigated in greater depth; we do not wish to draw conclusions on this until we manage to reduce the sparsity and consequently also the standard deviations of NMAE values.

Also interesting, the Cosine similarity works just as well as Pearson on EachMovie and Jester. Early researches show much poorer performance of the Cosine similarity measure (Breese et al. (1998)).

As a side-product we noticed that the true value of collaborative filtering (in general) is shown yet when computing NMAE over some top percentage of eccentric users. We defined eccentricity intuitively as MAE (mean absolute error) over the overlapping ratings between "the average user" and the user in question (greater MAE yields greater eccentricity). The average user was defined by averaging ratings for each particular item. This is based on the intuition that the ideal average user would rate every item with the item's average rating. The incorporation of the notion of eccentricity can give the more sophisticated algorithms a fairer trial. We computed average per-user NMAE only over the top 5% of eccentric users. The power of the kNN algorithms over Popularity became even more evident. In near future, we will define an accuracy measure that will weight per-user NMAE according to the user's eccentricity, and include it into our evaluation platform. We will also consider ways of handling the more eccentric users differently.

Acknowledgements

This work was supported by the Slovenian Ministry of Education, Science and Sport and the IST Programme of the European Community under SEKT Semantically Enabled Knowledge Technologies (IST-1-506826-IP) and PASCAL Network of Excellence (IST-2002-506778). This publication only reflects the authors' views. The EachMovie dataset was provided by Digital Equipment Corporation. The Jester dataset is courtesy of Ken Goldberg et al.. The authors would also like to thank Tanja Brajnik for her help.

References

1. BALDI, P., FRASCONI, P., and SMYTH, P. (2003): Modelling and Understanding Human Behavior on the Web. In: *Modelling the Internet and the Web, ISBN: 0-470-84906-1, 171–209*.
2. BREESE, J.S., HECKERMAN, D., and KADIE, C. (1998): Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In: *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*.
3. CLAYPOOL, M., LE, P., WASEDA, M., and BROWN, D. (2001): Implicit Interest Indicators. In: *Proceedings of IUI'01*.
4. DEERWESTER, S., DUMAIS, S.T., and HARSHMAN, R. (1990): Indexing by Latent Semantic Analysis. In: *Journal of the Society for Information Science, Vol. 41, No. 6, 391–407*.
5. GOLDBERG, K., ROEDER, T., GUPTA, D., and PERKINS, C. (2001): Eigentaste: A Constant Time Collaborative Filtering Algorithm. In: *Information Retrieval, No. 4, 133–151*.

6. GRCAR, M. (2004): User Profiling: Collaborative Filtering. In: *Proceedings of SIKDD 2004 at Multiconference IS 2004, 75–78.*
7. HERLOCKER, J.L., KONSTAN, J.A., TERVEEN, L.G., and RIEDL, J.T. (2004): Evaluating Collaborative Filtering Recommender Systems. In: *ACM Transactions on Information Systems, Vol. 22, No. 1, 5–53.*
8. HOFMANN, T. (1999): Probabilistic Latent Semantic Analysis. In: *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence.*
9. KONSTAN, J.A., MILLER, B.N., MALTZ, D., HERLOCKER, J.L., GORDON, L.R., and RIEDL, J. (1997): GroupLens: Applying Collaborative Filtering to Usenet News. In: *Communications of the ACM, Vol. 40, No. 3, 77–87.*
10. MELVILLE, P., MOONEY, R.J., and NAGARAJAN, R. (2002): Content-boosted Collaborative Filtering for Improved Recommendations. In: *Proceedings of the 18th National Conference on Artificial Intelligence, 187–192.*
11. RESNICK, P., IACOVOU, N., SUCHAK, M., BERGSTROM, P., and RIEDL, J. (1994): GroupLens: An Open Architecture for Collaborative Filtering for Netnews. In: *Proceedings of CSCW'94, 175–186.*