A Review of Recommender Systems using Collaborative Filtering

Pengfei Gao, s1144374

January 19, 2012

Abstract

The review presents an overview of the recent research area of recommender systems and describes the recent approaches based on the following three main types: content-based, collaborative filtering, and hybrid model. The review mainly focuses on the collaborative filtering approach and describes different algorithms implemented based on the two main models: memory-based model and model-based model. The review also introduces the current research on the time-drifting effects for the recommender systems.

Contents

1 Introduction

With the flourishing development of Internet, it is driven by the same belief in the unbounded opportunity to analyse the huge amount of data produced by people surfing the Internet. Online service providers and retailers are facing the challenges predicting users preferences and recommending the right, personalized contents that meet their tastes[1, 8, 20]. So recommender systems becomes an important issue in both research and industry areas. Recommender systems compares the items and users' information and find correlations among them, and provide a personalized and ranked items or content to users.

[1] presents an overview of current recommender systems strategies which are classified into three mainly approaches: content-based, collaborative filtering and hybrid models. The hybrid models combine the content-based approach and collaborative filtering and utilize them in different phases to avoid specific limitations of these two approaches. In content-based approaches, it first creates a profile for each user which describes user's tastes, preferences and requirements estimated by user's registered information, feedback and past behaviours. Each item also has a profile which contains its categories and characteristics which are usually represented as keywords extracted from the content of itself [2]. Some weighting measure may used to distinguish the importance of user and items' features [1]. Then calculate the relations among the users and items and recommend the best-matched items to users. For example, using cosine similarity measure to utilize the similar features among user and items' profile [20]. However, [1] indicates this approach require extracting features from items' contents which might not be easy.

The alternative approach to content-based approach is collaborative filtering which relies on users' past behaviours, such as item ratings and past transactions [17]. Collaborative filtering learns predictive models of user preferences and behaviours from historic data which is content domain free and avoids large amount of data collection[13]. [18] indicates that it allows complicated and unknown data patterns which might be difficult to characterise using known data features.

2 Collaborative Filtering

[7] identifies two general classes for collaborative filtering: memory-based

(also known as neighborhood models [13, 18]) and model-based. Memory-based methods are centred on calculating the relationships between users or items maintained in a database and the computation is performed across the whole database [24]. For example, an user-oriented approach predicts the rating $\hat{r}$ for user i to an item j based on the rating made by other users on the same item j. [7] presents several functions to explain this approach in details. Suppose S is the set of items and n is the total number of items set S that the user has rated before, then we can get the mean rating for the user u as:

$$\bar{r}_i = \frac{1}{n} \sum_{j \in S} r_{i,j} \tag{1}$$

We then can calculate the predicted ratings for user i based on some extra information about the user and some weights derived according to the user's preferences, profiles, and past behaviours. So for user i, we predict the rating $\hat{r}_{i,j}$ the user may rate on item j is computed as:

$$\hat{r}_{i,j} = \bar{r}_i + \kappa \sum_{t=1}^{n} w(i,t)(r_{t,j} - \bar{r}_i) \tag{2}$$

where n is the number of all users and t represents for each user in the sum loop. The weight function $w(i,t)$ may indicate distance, correlation, or similarity between other users and the current user i [7]. κ is a normalizing constant which can help the weights' absolute values sum to unity [7]. There are many ways to define the weight functions. One of the most famous statistical way was first introduced in [25] based on the Pearson correlation coefficient. The correlation between user i and user t is calculated as:

$$w(i,t) = \frac{\sum_{j \in S}(r_{i,j} - \bar{r}_i)(r_{t,j} - \bar{r}_t)}{\sqrt{\sum_{j \in S}(r_{i,j} - \bar{r}_i)^2 \sum_{j \in S}(r_{t,j} - \bar{r}_t)^2}} \tag{3}$$

where the item j was rated both by user i and user j before. The weights scales between -1 and 1 indicating how similar of the ratings made by user i and user t on the same items.

Memory-based methods are very popular and have been used in many areas, for example GroupLens project[25] and Amazon[20], because of its simplicity and intuitive process which avoid from building data analysis and modelling stage[13]. However, it is relying on a few specific neighbourhood relationships such as the above user-oriented approach which ignored the characteristics and uniqueness of a specific user to distinguish with other users. Hence, nothing is really learned from the existed users' profiles, preferences and past behaviours which may cause problems when adding new items having no ratings or new users haven't rating anything now. Moreover, for massive datasets, memory-based methods relies on the expensive offline computation which will result in the bad end-user experience for online users.

Model-based approaches have overcomes the drawbacks of memory-based methods. This method first tries to understand users from their profiles, preferences and past behaviours and cluster them into different models, and then calculate the predicted ratings for users on new items. [7] presents a probabilistic model for collaborative filtering using a Bayesian classifier to cluster ratings depend on some given information. Users who have common interests and tastes may rating same items in a similar way and those users may be grouped into a specific class. Suppose the ratings are discrete number in a limited range, we set the range is from 0 to m here:

$$p_{i,j} = E(r_{i,j}) = \sum_{k=0}^{m} P(r_{i,j} = k | r_{i,l}, l \in S_i) \times k \tag{4}$$

where $p_{i,j}$ is the probability of a particular rating for user i may rate on the item j given the user's past recorded ratings $r_{i,l}, l \in S_i$, S_i is the item set containing all the items that user i has rated before. [27] describes a probabilistic model using cluster methods to separate both users and items into different clusters, such as EM algorithms, repeated K-means clustering, and Gibbs sampling. [7] proposes a naive Bayes method to assign users c into different clusters C:

$$P(C = c, r_1, ..., r_n) = P(C = c) \prod_{i=1}^{n} P(r_i | C = c) \tag{5}$$

where $r_1, ..., r_n$ is the existed ratings they may rate on $1, ..., n$ items and $P(C = c, r_1, ..., r_n)$ is the probability of users belonging to the particular cluster c. The conditional probabilities $P(r_i|C = c)$ is estimated based on the ratings made by other users belonging to the cluster c. After that, [7] uses Cheeseman and Stutz (1995) to find the efficient approximations for the marginal likelihood to assign the users or items to the particular clusters.

3 Probabilistic Latent Semantic Analysis

Hofmann proposes a statistical approach for collaborative filtering named probabilistic Latent Semantic Analysis (pLSA) [12, 14, 15]. It first make user-item pairs (i, j) of all possible users $i \in U$ and items $j \in S$ in the databases. Then hidden variables Z are introduced to describe the relationships derived by modelling the joint distribution of users and items as a mixture distribution [8]. Hofmann wrote the mixture model as the following equation:

$$p(j|i;\theta) = \sum_{z} P(j|z)P(z|i)$$

The parameters θ represent the conditional probabilities $p(z|i)$ and $p(j|z)$. The hidden variables Z are used to represent a reason for user i to prefer a item j. Each state $z \in Z$ is supposed to explain the unknown reason that can not be observed. So we can separate users into related user clusters and items into the related clusters based on those hidden variables Z. Then Hofmann estimates the model parameters θ such that the product of conditional log-likelihood is maximized:

$$L(\theta) = -\frac{1}{N}\sum_{n=1}^{N} logP(j_n|i_n;\theta)$$

where N is the size of user-item pairs. EM algorithm is used to estimate the maxmized log-likehood parameters. The E-Step computes the optimal Q , the posterior probabilities of the latent variables $\hat{\theta}$, given the parameter $\hat{\theta}$:

$$Q^{*}(z;i,j;\hat{\theta}) = P(z|i,j;\hat{\theta}) = \frac{\hat{P}(j|z)\hat{P}(z|i)}{\sum_{z' \in Z}\hat{P}(j|z')\hat{P}(z'|i)}$$

The M-Step is for the optimization of the parameters θ:

$$P(j|z) = \frac{\sum_{i} Q^{*}(z;i,j;\hat{\theta})}{\sum_{j}\sum_{i'} Q*(z;;i,j,\hat{\theta})}$$

$$P(z|i) = \frac{\sum_{j} Q^{*}(z;i,j;\hat{\theta})}{\sum_{z'}\sum_{j} Q^{*}(z';i,j;\hat{\theta})}$$

[8] introduces a way to perform the EM computation paralleled and scalable using the MapReduce model [9].

[1] points out a limitation of this method that each user will be assigned to the most likely cluster while the user may interest totally different and unrelated topics such as driving and reading.

4 Singular Value Decomposition

Due to the expensive computation cost of the huge increasing amount of data and the sparsity of the data becomes a series problem, lots of dimensionality reduction methods has been investigated both in the research and industry areas to address the problems. [6] proposes a approach using Singular Value Decomposition(SVD). First [6] constructs a big user-item matrix which contains boolean values indicating whether the user has rated the items. In fact, the matrix is really sparse because there may be millions of items while a user may only rate dozens of them. In other words, the majority values in the matrix will be zero. SVD is a famous matrix factorization method that will can decompose a $m \times n$ matrix A into the product of three matrices as following:

$$A = U \cdot \sum \cdot V^T$$

where U is a $m \times r$ matrix and V is a $n \times r$ matrix; r is the rank of the diagonal matrix $\sum$ which contains the singular values. The numerical values of those singular values reflects the impact they can affect how much captured of the original matrix A. For example, in a particular situation, k largest singular values with the corresponding singular vectors in the matrices of U and V can recover the important "latent" structure of the original matrix A [6]. Therefore, we can reduce the the size of singular values from r to k. Moreover, the user ratings and items can represented in k-dimensional space.

[26] presents two experiments on dimensionality reduction in recommender systems comparing the quality using SVD and collaborative filtering. They choose Mean Absolute Error (MAE) as the evaluation metric to compare these two methods. The result shows that SVD methods is much better and can provide better online recommendation performance than collaborative filter.

5 Explicit and Implicit Feedback

Recommender systems make predictions based on the different kinds of input. [16] indicates the high quality explicit feedback which directly reflects users' preference on the items contribute the most convenient to the recommending process. For example, Netflix records the star ratings for movies and Youtube uses "Like" and "Dislike" buttons as a vote for the items. However, in some practical situations, explicit feedback is not enough for an accuracy recommender system. For example, Google News [8] treats a user's click on an article as a positive vote. But, regarding a click as a positive vote is much more noisy than the Netflix explict 1-5 star ratings. For a web page within lots of news links, a user may make many wrong clicks on those news which the user do not want to read. What's more, as the clicks can be regarded as a user's interest, they can not say anything about a user's negative interest. Thus, derive users preference from users' implicit feedback such as observing user behaviours is a very important way to improve the accuracy of the recommender systems [22]. [16] lists 4 types of the implicit feedback such as purchase history, browsing history, search patterns, and even mouse movements. After collecting the feedback, the issue becomes how to process them. [16] indicates that it is very hard to identify the unique features of those various implicit feedback collected from various approaches directly using the algorithms designed for the explicit feedback. Google News uses a covisitation instances [8] to process the user clicks. Whenever Google receive a click from the user i on an article j, they will search the user's recent click history C_i and iterate over the items in it. For such articles $k \in C_i$, they create a pair (j, k) and add a weighted adjacency distance to the pair. If the pair has already been created, they will update the age discounted count. So for a given article j, its near neighbours are effectively the set of articles the user want to view.

6 Hybrid model

From the above, we can see that memory-based methods are better at modelling the neighbour relationships and really effective among the very small localized system while model-based methods are better at estimate the whole datasets and perform better for the larger scale systems. The Netflix prize has attracted a great of attention on the Collaborative Filtering area since 2006. With no team reached the perfect requirements of the prize, the prize was awarded the annual progress prize to the KorBell team which achieved the most outstanding improvement so far [5]. Koren, one of the two members in the KorBell team, proposed a hybrid model that improve accuracy and efficiency by utilizing the advantages of both memory-based and model-based methods. In [3–5, 16], Koren et al. present a baseline estimates to adjust the effects caused by some users who made higher ratings than other users and some items that are more popular than other items as following:

$$b_{i,j} = \mu + b_i + b_j \tag{6}$$

where b_i and b_j are the averages recorded ratings user i made and the item j received, respectively. $b_{i,j}$ is the baseline estimate predicted by the system for user i on the item j. Therefore, for the memery-based model, equation (2) is changed to the following form:

$$\hat{r}_{i,j} = b_{i,j} + \sum_{t \in S_i} (r_{i,j} - b_{i,j}) w_{i,j} + \sum_{t \in N_i} c_{j,t}$$

where $c_{j,t}$ is the global offsets similar the weight factor $w_{j,t}$ from item j to item t [5]. N_i is the implicit feedback observed from user i behaviour, for example, user i has searched the item for twenty times while he rarely search any other items more than three times. In order to predict faster and more specific recommendations, and also avoid overfitting problems for new users who rarely rate, Koren moderates the equation:

$$\hat{r}_{i,j} = b_{i,j} + |R^k_{(j;i)}|^{-\frac{1}{2}} \sum_{t \in R^k_{(j;i)}} (r_{i,t} - b_{i,t}) w_{j,t} + |N^k_{(j;i)}|^{-\frac{1}{2}} \sum_{t \in N^k_{(j;i)}} c_{j,t} \qquad (7)$$

where $R^k_{(j;i)} = R_i \cap S^k_j$ and $N^k_{(j;i)} = N_i \cap S^k_j$. S^k_j is the dataset of k items that are most similar to item j.

For model-based methods, [18] presents a matrix factorization models which is much similar to the SVD for identify latent factors of the ratings matrix as following:

$$\hat{r}_{i,j} = b_{i,j} + p_i^T q_i \qquad (8)$$

where f is the dimension of a joint latent factor space. Each user i links to a user-factors vector $p_i \in \mathbb{R}^f$ and each item j to an item-factors vector $q_j \in \mathbb{R}^f$. [17] indicates that for the given user i, the factors of p_i measure the degree of the user prefers the items on the relating factors. The above equation reveals the interaction between the user i and the item j - the user's preference for the item's features. Following the gradient descent optimization performed in [23], [18] proposes a rule:

$$\hat{r}_{i,j} = b_{i,j} + q_j^T (|R_i|^{-\frac{1}{2}} \sum_{t \in R_i} (r_{i,t} - b_{i,t}) x_t + |N_i|^{-\frac{1}{2}} \sum_{t \in N_i} y_t) \qquad (9)$$

where $q_j, x_t, y_t \in \mathbb{R}_f$ are factor vectors relating to the each item j. In the other way, we can define users using the items they intrested in without providing the explicit parameterization for users. Therefore, the previous user factor p_j can be replaced by $|R_i|^{-\frac{1}{2}} \sum_{t \in R_i} (r_{i,t} - b_{i,t}) x_t + |N_i|^{-\frac{1}{2}} \sum_{t \in N_i} y_t$. [18] called this model "Asymmetric-SVD". This new model reduces the parameters from the number of users to the number of items that the user prefer recorded by both explicit and implicit feedback. Thus, it lowers the complexity of the computation. The second, since there is no parameterization for users in the Asymmetric-SVD model, the system will recommend services to users instantly once they provide any kind feedback. In other words, the system can immediately utilize the new ratings to update user preferences. The third, [11] indicates the fact that a user may be willing to risk buy a book based on the recommendation systems while he will not risk choosing a vacation travel recommended by such a system. The reason is that the system make recommendations based on the computation of large amount of sparse and incomplete data in a black box. There is no explanation capabilities to persuade users to belief in. Users will not trust the system unless they know the reasons why the system recommend the items. However, the predictions made by the Asymmetric-SVD model are straightforwardly based on users' past feedback. Moreover, the model can allow the user to identify which of the other user's past feedback influence the predictions most. Koren combines these two models: the memory-based model and the model-based model, by summing the equations of (6) and (7). Therefore, these two models nicely complement each other as follows:

$$\hat{r}_{i,j} = b_{i,j} + q_j^T(|R_i|^{-\frac{1}{2}} \sum_{t \in R_i} (r_{i,t} - b_{i,t}) x_t + |N_i|^{-\frac{1}{2}} \sum_{t \in N_i} y_t)$$

$$+ |R^k_{(j;i)}|^{-\frac{1}{2}} \sum_{t \in R^k_{(j;i)}} (r_{i,t} - b_{i,t}) w_{j,t} + |N^k_{(j;i)}|^{-\frac{1}{2}} \sum_{t \in N^k_{(j;i)}} c_{j,t} \qquad (10)$$

The above equation presents a 3-level model for predictions. The first level, $b_{i,j}$ (6), describes the normal properties of the user and item without considering any relations between them. For example, ipod is an electronic product and the user's rating scale is the average. The second level, $q_j^T(|R_i|^{-\frac{1}{2}} \sum_{t \in R_i}(r_{i,t} - b_{i,t})x_t + |N_i|^{-\frac{1}{2}} \sum_{t \in N_i} y_t)$ considers the relation between the user properties and item properties. For example, ipod receives good ratings and the user prefer rating high on new and cool electronic products. The last neighbourhood level is focus on the grained adjustments that is special and not easy found on user's properties. For example, the user rated very low on all the products made by Apple. Above all, the Asymmetric-SVD model presents a straightforwardly explanation on the predictions made by the system and avoid the re-training the model for adding new items or users. The explainability will win the confidence of the users and the hybrid model improves the accuracy and efficiency by addressing different features of the original data.

7 Temporal Dynamics

So far, the models describes here are all static. In reality, items' perception and popularity change constantly due to the changes external circumstances. For example, Sony Walkman was very popular but now it was disappearing as electronic music player becomes the mainstream of the market. Similarly, customers' inclinations evolve leads people to redefine their preferences and interests. Thus, recommendation should involve the temporal effects which reflect the time-drifting, dynamic nature of user-items interactions [10]. Koren [19] proposes a method to model the time changing behaviour over the life span of the original data and applies to the recommender systems based on the matrix factorization approach [17]. The method models temporal effects by decomposing user ratings into distinct aspects which vary over time: item biases $b_j(t)$, user biases $b_i(t)$, and user preferences $p_i(t)$. The item biases $b_j(t)$ reflects the item j popularity variation over time and the user biases $b_i(t)$ reflects user i rating baselines variation over time. The user preferences $p_i(t)$ reflects the changes of user's preferences over time. So, the exact parametrizations of time-drifting parameters change the equation (6) and equation (8) into the following:

$$\hat{r}_{i,j}(t) = \mu + b_j(t) + b_i(t) + q_j^T p_i \tag{11}$$

Koren applied the method into collaborative filtering recommender approaches and the results proved the temporal dynamic very useful to improve the quality of the recommendations [19] .

8 Conclusion

As described above, there has been lots research done in both academic and industry areas on the recommender systems during the past years. A broad range of techniques such as statistical, information retrieval, machine learning and others have been applied to the recommender systems. As shown before, the current recommender systems can be categorized into three main types: content-based, collaborative filtering, and hybrid model. Content-based methods classify a series of discrete features of the items in order to make recommendations which share the similar properties. The collaborative filtering model focus on learns the predictive models of user preferences from the historic data, such as users' past behaviours and profiles.

Memory-based and model-based models are the two general classes for collaborative filtering. Memory-based methods focus on calculating the localized relationships between users and items and the computation is performed across the whole database. Memory-based methods are very popular because of its simplicity and intuitive process which avoid from building data analysis and modelling stage. The expensive computation over the whole dataset and the ignorance the characteristics and uniqueness of a specific user form other users. Model-based methods try to build models for users from their profiles, preferences and past behaviours and cluster them into different classes. The model-based techniques includes naive Bayesian clustering techniques, Bayesian networks, dependency networks and matrix factorization. In the real life the users' feedback has been classified into two classes. One is the high quality explicit feedback and the other is the implicit feedback collected by such as observing user behaviours. As both memory-based and model-based methods has their own limitations, the hybrid approach by combining the these two models can help to improve the accuracy and efficiency by utilizing the strengths of both them. The review introduces the matrix factorization approach proposed by Koren et al. [17] in details. The complex factor dimensionality which descriptions contains more distinct parameters can help to improve the accuracy. Moreover, the experience such as Netflix Prize [5] has shown that matrix factorization approach performance is beyond the classical recommendation techniques. At last, the review briefly introduces the temporal effects which reflect the time-drifting nature of user-items interactions and the methods modelling the time changing behaviour over the life span of the original data.

Nowadays, most research on the recommender systems has focused on improving the accuracy of the algorithms. However, being accurate is not enough [21]. Moreover, the testing and evaluation work of the most recommender systems is based on the historic data, how to find a meaningful and credible recommendation for users is more important and valuable than the accuracy. The research of recommender systems should do more from a user-centric perspective not only efficient and accurate but also meaningful and trustful.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734 – 749, june 2005.

[2] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40:66–72, March 1997.

[3] R.M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43 –52, oct. 2007.

[4] Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 95–104, New York, NY, USA, 2007. ACM.

[5] Robert M. Bell and Yehuda Koren. Lessons from the netflix prize challenge. *SIGKDD Explor. Newsl.*, 9:75–79, December 2007.

[6] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 46–54, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[7] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In Gregory F. Cooper and Serafín Moral, editors, *UAI*, pages 43–52. Morgan Kaufmann, 1998.

[8] Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 271–280, New York, NY, USA, 2007. ACM.

[9] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[10] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 485–492, New York, NY, USA, 2005. ACM.

[11] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[12] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 50–57, New York, NY, USA, 1999. ACM.

[13] Thomas Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 259–266, New York, NY, USA, 2003. ACM.

[14] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22:89–115, January 2004.

[15] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI '99, pages 688–693, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[16] Yifan Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 263 –272, dec. 2008.

[17] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30 –37, aug. 2009.

[18] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 426–434, New York, NY, USA, 2008. ACM.

[19] Yehuda Koren. Collaborative filtering with temporal dynamics. *Commun. ACM*, 53:89–97, April 2010.

[20] G. Linden, B. Smith, and J. York. Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76 – 80, jan/feb 2003.

[21] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *CHI '06 extended abstracts on Human factors in computing systems*, CHI EA '06, pages 1097–1101, New York, NY, USA, 2006. ACM.

[22] Douglas W Oard and Jinmook Kim. *Implicit Feedback for Recommender Systems*, pages 81–83. 1998.

[23] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. *Statistics*, 2007:2–5.

[24] David M. Pennock, Eric Horvitz, Steve Lawrence, and C. Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI*, pages 473–480. Morgan Kaufmann, 2000.

[25] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[26] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *IN ACM WEBKDD WORKSHOP*, 2000.

[27] Lyle Ungar, Dean Foster, Ellen Andre, Star Wars, Fred Star Wars, Dean Star Wars, and Jason Hiver Whispers. Clustering methods for collaborative filtering. AAAI Press, 1998.