Analysis of Densification on Implicit Feedback Sparse Datasets

Preeti Lekha
School of Computer Science
Oklahoma State University
Stillwater,Oklahoma-74075,USA
Email:preeti.lekha@okstate.edu

Christopher Crick
Computer Science Department
Stillwater,Oklahoma-74075,USA
Telephone:(405)-744-5673
Email:chriscrick@cs.okstate.edu

***Abstract*— Given the large number of items (web-pages, videos) available on the Web, users benefit from being shown only items of potential interest to them. One of the tools that address this challenge is recommender systems. Recommendation Systems apply Information Retrieval techniques to select the online information relevant to a given user. Collaborative Filtering (CF) is currently most widely used approach to built Recommendation System. CF techniques use the user's behavior in form of user-item ratings as their information source for prediction. There are major challenges like the sparsity of the rating matrix and the growing nature of data which are faced by CF algorithms. These challenges have been well addressed by Matrix Factorization (MF). In this work we focus on studying the collaborative filtering algorithm, Matrix Factorization, considered to be a state-of-the-art approach, and summarize our experiences and lessons learned from experiments on the implicit feedback data domain. Specifically, we suggest that not just the characteristics of a domain (e.g., close connections versus loose connections among users) but also the density(e.g., density of the feedback matrix) property of a dataset helps in predicting the performance of Matrix factorization approach used for a particular recommendation problem.**

***Index Terms*—Collaborative Filtering, Recommendation Systems, Neighborhood Approaches, Matrix Factorization, Implicit Feedback.**

I. INTRODUCTION

The growth of the Internet over the past decade has resulted in an exponential growth of the online content. With such large volumes of data, one of the challenges for a Web user is finding the right content given his or her interests. For example, users may be interested in satisfying a current information need, or finding a product of interest (such as a movie, news, etc.). While search engines partially alleviate the problem of finding content from the Web, especially when an information need can be expressed as a query, in many cases, a user may not be aware of what to look for. Recommender systems can be used to address this problem, by suggesting to users items that the users are most likely to interact with, but perhaps would not find in the overwhelming number of available items. These systems provide users with personalized recommendations for products or services, which hopefully suit their unique taste and needs. The technology behind those systems is based on profiling users and products, and finding how to relate them. Recommender systems have been widely studied in both academia and industry in the last decade because of abundant application domains and significant room remains for improvement in personalization accuracy.

Broadly speaking, recommender systems are based on two different strategies (or combinations thereof). The content based approach creates a profile for each user or product to characterize its nature. As an example, a movie profile could include attributes regarding its genre, the participating actors, its box office popularity, etc. User profiles might include demographic information or answers to a suitable questionnaire. The resulting profiles allow programs to associate users with matching products. However, content based strategies require gathering external information that might not be available or easy to collect.

An alternative strategy relies only on past user behavior without requiring the creation of explicit profiles. This approach is known as Collaborative Filtering (CF), a term coined by the developers of the first recommender system - Tapestry [11]. Tapestry was an electronic messaging system that allows users to rate messages good or bad or associate text annotations with those messages. In a recommendation application, a CF system tries to find other like-minded users and then recommends the items that are most liked by them based on opinions of other users. CF analyzes relationships between users and interdependencies among products, in order to identify new user-item associations. For example, some CF systems identify pairs of items that tend to be rated similarly or like-minded users with similar history of rating or purchasing to deduce unknown relationships between users and items. The only required information is the past behavior of users, which might be their previous transactions or the way they rate products. Collaborative Filtering is the most popular approach to build Recommendation System and has been successfully employed in many applications. The CF recommender system works by collecting user feedback in the form of ratings for items in a given domain [22]. The most common types of CF systems are user-based and item-based approaches. The key advantage of a CF recommender system is that it does not rely on the machine analyzable contents and therefore it is capable of accurate recommendations. In CF, users who had similar choices in the past will have similar choices in the future as well. Another major appeal of CF is that it is domain free, yet

it can address aspects of the data that are often elusive and very difficult to profile using content based techniques. While generally being more accurate than content based techniques, CF suffers from the cold start problem, due to its inability to address products new to the system, for which content based approaches would be adequate.

Early collaborative filtering algorithms for recommendation systems utilize association inferences, which have a very high time complexity and a very poor scalability. Recent methods that use matrix operations are more scalable and efficient. The implementations and algorithms of collaborative filtering for the applications of recommendation systems face several challenges. First is the size of processed datasets. The second one comes from the sparseness of rating matrix, which means for each user only a relatively small number of items are rated. So, these challenges are well addressed by Matrix Factorization [22][14].

Matrix Factorization (MF) plays an important role in the Collaborative Filtering recommender system. MF has recently received greater exposure, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction [14][21]. Prediction of ratings and Recommendations can be obtained by a wide range of algorithms, while Neighborhood-based Collaborative Filtering methods are simple and intuitive. The Matrix Factorization techniques are usually more effective because they allow use to discover the latent features underlying the interactions between users and items. Matrix Factorization is simply a mathematical tool for playing around with matrices, and is therefore applicable in many domains where one would like to find out something hidden under the data.

Recommender systems rely on different types of input. Most convenient is the high quality explicit feedback, which includes explicit input by users regarding their interest in products. For example, Netflix collects star ratings for movies and TiVo users indicate their preferences for TV shows by hitting thumbs-up/down buttons. However, explicit feedback is not always available. Thus, recommenders can infer user preferences from the more abundant implicit feedback, which indirectly reflect opinion through observing user behavior [19]. Types of implicit feedback include purchase history, browsing history, search patterns, or even mouse movements. For example, a user that purchased many books by the same author probably likes that author. The vast majority of the literature in the field is focused on processing explicit feedback, probably thanks to the convenience of using this kind of pure information. However, in many practical situations recommender systems need to be centered on implicit feedback. This may reflect reluctance of users to rate products, or limitations of the system that is unable to collect explicit feedback. In an implicit model, once the user gives approval to collect usage data, no additional explicit feedback (e.g. ratings) is required on the users part.

In this work we aim to investigate the performance change with the increasing density/reducing sparsity of a dataset. We focus on applications with large datasets and with implicit user feedback (as this type of data is more common in real-world applications and less explored in the literature). To perform our study, we use a state-of-the-art CF algorithm, specifically Matrix Factorization, a global model-based approach, given that this algorithm can capture different types of information.

The Matrix Factorization (MF) algorithm, specifically a variant that was designed to handle implicit feedback [12], is a latent factor approach and captures user similarity globally by representing all users in a common latent space.

II. RELATED WORK

Given the diverse area of recommender systems, we only review works most relevant to our study. Collaborative Filtering Approaches: CF is a popular and widely used approach to recommender systems, regardless of the application domain. The basic idea of CF-based algorithms is to provide item recommendations based on the opinions of other like-minded users [10], [23]. CF approaches can be grouped into neighborhood-based approaches and latent factor model-based approaches [8], [25]. Neighborhood-based techniques [4], [10], [23], [24], [16] predict the preference of a user for an item based on the preferences given to that item by neighbor users, and have the following merits: ability to explain recommendations to a user, easy and incremental addition of new data, small number of parameters to tune, and their intuitiveness [6]. Different from the neighborhood-based approaches, latent factor model-based techniques [5], [12], [14] learn a predictive model from the user-item preference information to predict unknown user-item preferences [3], [8] and have the advantage of better recommendation accuracy compared to the neighborhood-based approaches at the expense of higher computational cost [12], [12], [23]. Furthermore, some approaches are designed to handle explicit user feedback (ratings, like/dislike) in the data domains [5], [10], [14], [23], [24], while other approaches assume implicit user feedback (clicks, buys) in the data domains [4], [12], [16].

Relevant Applications: In the past decade, recommender systems have been used to address the information overload problem in many application domains. For example, Konstan et al. [13] proposed several collaborative filtering approaches to recommend usenet news, Linden et al. [16] proposed item-to-item collaborative filtering techniques to recommend items in Amazon and Baluja et al. proposed the Adsorption technique to recommend YouTube videos. Several approaches based on matrix factorization were proposed to predict movie ratings as part of the Netflix Challenge [5], [12], [14]. Among many other approaches to the Netflix problem, Bell et al. [5] proposed a neighborhood-based technique which combines k-nearest-neighbor (kNN) and low-rank approximation to obtain significantly better results compared to either technique alone. Their team won the progress prize in October 2007, obtaining an RMSE score on the qualifying dataset of 0.8712, improving the CineMatch score by 8.5. However,

their solution [7] is a linear combination of 107 individual solutions, while multiple solutions are derived by variants of three classes of solutions (ALS, RBM, and kNN). For ALS alone, their best result was obtained using 128 hidden features with an RMSE score above 0.9000. Furthermore, Yang et al. [26] addressed the co-author recommendation problem in the DBLP co-author network as a link prediction problem in a heterogeneous network. Wang et al. [25] proposed an approach which combines textual data from scientific articles and the list of papers in user libraries to recommend scientific articles to users of online communities.

Although various approaches have been extensively used for different recommendation applications in the literature, we have found no prior work that studied Matrix Factorization performance with the change is density of sparse dataset, and what knowledge and characteristics about the domain can help predict the performance of MF.

III. RECOMMENDATION ALGORITHM

In this section, we will briefly explain the Matrix Factorization algorithms used in this work.

A common problem faced by internet companies is that of recommending new products to users in personalized settings. This can be formulated as a learning problem in which we are given the ratings that users have given certain items and are tasked with predicting their ratings for the rest of the items. Formally, if there are *n* users and *m* items, we are given an $n \times m$ matrix *R* in which the $(u, i)^{th}$ entry is r_{ui} - the rating for item *i* by user *u*. Matrix *R* has many missing entries indicating unobserved ratings, and our task is to estimate these unobserved ratings. A popular approach for this is matrix factorization.

Matrix Factorization (MF) is a general term describing a family of algorithms suitable for CF. Collaborative filtering is popular and widely deployed in Internet companies like Amazon [16], Netflix [1], Google News [9], and others. The intuition behind MF is that it is possible to find some latent aspects that can explain users preferences for items. Several MF approaches have been proposed in the literature to decompose the user-item rating matrix, and subsequently use the latent user and item factors to recommend items to users [6], [14]. Hu et al. [12] pointed out some unique characteristics of implicit feedback that restrict the usage of traditional MF approaches, and proposed a novel decomposition by introducing preference and confidence variables into the factorization. The idea is to associate small confidence values even if the user did not prefer an item, since not preferring an item may stem from various reasons such as lack of knowledge about the item or limited availability, beyond not liking it. In this work, we have used the MF approach proposed in [27] as our datasets have implicit user feedback. Please refer to the works in [3], [5], [14] for a general introduction to MF approaches.

A. ALS Matrix Factorization

In this work we have used Alternating least squares Matrix Factorization [12] [17]. Alternating-least-squares Matrix Factorization is a parallel algorithm involving alternating-least-squares with weighted-λ-regularization (ALS-WR). ALS matrix factorization algorithm is parallelized and optimized to scale up well with large, sparse data. Alternating least squares (ALS) is a powerful MF algorithm for both explicit and implicit feed-back based recommender systems. As shown in many articles, increasing the number of latent factors (denoted by K) boosts the prediction accuracy of MF based recommendersystems, including ALS as well. The price of the better accuracy is paid by the increased running time. Yet, the running time of model building can be important in recommendation systems; if the model cannot keep up with the changing item portfolio and/or user prole, the prediction accuracy can be degraded.

B. ALS with Weighted--Regularization

As the rating matrix contains both signals and noise, it is important to remove noise and use the recovered signal to predict missing ratings. *Singular Value Decomposition* (SVD) is a natural approach that approximates the original user-item rating matrix *R* by the product of two rank-*k* matrices $\widehat{R} = U^T \times M$. The solution given by the SVD minimizes the Frobenious norm of *R* - $\widehat{R}$, which is equivalent to minimizing the RMSE over all elements of *R*. However, as there are many missing elements in the rating matrix *R*, standard SVD algorithms cannot find *U* and *M*. [27] use alternating-least-squares (ALS) to solve the low-rank matrix factorization problem as follows:

1) Initialize matrix *M* by assigning the average rating for that co-author as the first row, and small random numbers for the remaining entries.
2) Fix *M*, Solve for *U* by minimizing the objective function
$$\pounds_{\lambda}^{reg}(R, U, M) = \pounds^{emp}(R, U, M) + \lambda(\|U\Gamma_U\|^2 + \|M\Gamma_M\|^2) \quad (1)$$
where Γ_U and Γ_M are Tikhonov matrices, *U* = $[u_i]$ is the user feature matrix, *M* = $[m_j]$ is the item feature matrix, *R* is the user-item matrix.
3) Fix *U*, solve for *M* by minimizing the objective function similarly;
4) Repeat Steps 2 and 3 until a stopping criterion is satisfied.

Observe that when the regularization matrices (U, M) are non-singular, each of the problems of Steps 2 and 3 of the algorithm has a unique solution. Also the sequence of achieved errors $L_{reg}^{\lambda}(R, U, M)$ is monotone non-increasing and bounded, hence this sequence converges. Rather than going all the way to convergence, [27] use a stopping criterion based on the observed RMSE on the probe dataset. After one round of updating both *U* and *M*, if the change in RMSE on the probe dataset is less than 1*bps*, the iteration stops and we use the obtained *U*, *M* to make final predictions on the test dataset. There are many free parameters. Without regularization, ALS might lead to overfitting. A common fix is to use Tikhonov regularization, which penalizes large parameters. [27] tried various regularization matrices, and eventually found the following weighted--regularization to work the best, as it

never overfits the test data (empirically) when we increase the number *nf* of features or the number of ALS iterations.

We use the following notation in this paper. N and M denote the number of users and items respectively. Latent factor dimension (D) is set to 20 by default, if not explicitly specified. In this work number of ALSMatrix-Factorization iterations is set to 5. Also $u\epsilon\{1...N\}$ as index for users, and $i, j\epsilon\{1...M\}$ as indices for items. The rating of user *u* on item *i* is r_{ui}, and its prediction is $\widehat{r}_{ui}$. All r_{ui} ratings are arranged in **R**. R^+ denotes each (u, i) pair of **R** i.e $R^+ = N.M$

To evaluate the ALSMatrix-Factorization performance on DBLP co-author dataset, root-mean-square error (RMSE) measure is used. RMSE is a standard statistical metric to measure model performance .

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij} - R^*_{ij})^2}{\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}}} \quad (2)$$

IV. EXPERIMENTAL DESIGN

Datasets and Preprocessing: We have used implicit feedback dataset to study the performance of Matrix Factorization proposed in [27]. The dataset used is the DBLP co-author dataset [2], [15]. The dataset has information about user-user collaborations between years 1940 and 2013, and the task is to recommend new collaborations. The dataset consists of approximately 1.3M users (who can also be seen as items) and 18.9M collaboration records with four columns, specifically, from id, to id, weight, and timestamp. Given the small median for the number of items clicked, 3, this dataset is seen as a sparse dataset.

Since we are not focusing on analyzing the impact of long user history or short user history, we do not consider the timestamp column of the DBLP co-author dataset. We used the graphlab implementation of matrix factorization version [17] which assumes there is a single rating between each user and data item. To achive this, whenever there are multiple instances of the same user-item pair, with the same ratings, they are pruned and only one is left with the weight column now modified to number of multiple instances. Multiple instances of user-item pairs indicate the confidence value between the user-item pair. The test data contains the following format: [author] [coauthor] [confidence value] [time]. It has 1,314,050 vertices (authors), total 18,986,618 edges (collaborations) of which 10,724,828 edges (collaborations) are unique. Average degree of graph is 28.898 $edges/vertex$ with the overall density of $1.2422 \times 10^{-5} edges/vertex^2$.

Experiments: We run Matrix Factorization on the processed dataset and evaluate the performance based on the RMSE value generated, to address the following questions:

1) **Given a large-scale data domain with a sparse user-item preference matrix,is MF approach is preferable?** Previous work [4] suggests that given that Adsorption is a neighborhood approach and captures local neighborhood information, it performs better on datasets with strong local-neighbourhood. Also [20] suggests that given the strong local-neighbourhood in DBLP co-author dataset, Adsorption performs better for the DBLP co-author dataset. In this work we want to experiment and observe if ALSMatrix-Factorization gives us any better performance as compared to results presented in [20].
2) **How the performance of the algorithm varies on a sparse-dataset with continuous densification.**
In our experiment we densify the DBLP co-author dataset by 10 percent in each iteration by dynamically adding more connections, and monitor the impact of density on the RMSE value of the resulting model. We hypothesize that since the DBLP co-author dataset has strong local neighborhood, the performance of Alternating-least-squares Matrix Factorization improves on densification of the DBLP co-author dataset.
3) **Does densification impacts the computational complexity in terms of running time of the algorithm.** We hypothesize that since densification of DBLP co-author dataset increases the amount of data that needs to be processed, the time complexity of the algorithm would increase to provide accurate results.
4) **What characteristics of a sparse dataset determine the performance of Matrix Factorization algorithm in a given domain.**
We believe that knowledge about how links are formed in the domain (based on close connections - resulting in strong local neighborhoods, or based on loose connections - resulting in useful global information) is important to decide which CF approach to use when not much data is available. In Sparse datasets, MF algorithms perform better when the links are formed dynamically, as it captures global user information through latent factors. But in case of the DBLP co-author dataset, links are formed based on close connections - resulting in strong local neighborhoods. We hypothesize that initially the performance may not be as good, but the performance goes on improving as the dataset is densified dynamically.

V. RESULTS

The analysis of results for the DBLP co-author dataset (Table I), along with a discussion of research questions, is performed below.

When comparing RMSE resulting from ALSMatrix-Factorization, the results from Table I support our hypothesis that ALSMatrix-Factorization does perform well even though the DBLP co-author sparse dataset has strong local-neighborhood and has previously generated good results with Adsorption Algorithm as per [20]. By choosing the Matrix-Factorization algorithm carefully for sparse datasets with strong local-neighborhood, a good recommender model can be built.

In the experiment, we also notice that as we increase

Experimental Results		
Density(p)	**RMSE**	**Runtime(T in ns)**
1.2422×10^{-5}	0.410746860859824	116273232000
1.3664×10^{-5}	0.39641893908558584	119360578000
1.4906×10^{-5}	0.38457202640625066	119580715000
1.6148×10^{-5}	0.3742318794594911	135149159000
1.7391×10^{-5}	0.3648199614395946	136568704000
1.8633×10^{-5}	0.35742650632774564	143724972000
1.9875×10^{-5}	0.34926164797852527	160485225000
2.1117×10^{-5}	0.0.34357677887060395	161851130000

Table I

RMSE SCORES FROM ALS-MATRIX-FACTORIZATION FOR THE DBLP CO-AUTHOR DATASET . THE DEMENSIONALITY(D) AND NO OF ITERATIONS (K) ARE SET TO 20 AND 5, RESPECTIVELY. IN THE TABLE, P INDICATES THE DENSITY OF DATASET IN TERMS OF FILL($edges/vertex^2$). RMSE INSDICATES THE ROOT-MEAN-SQUARE-ERROR. AND RUNTIME(T)INDICATES THE RUNTIME OF ALS-MATRIX-FACTORIZATION FOR VARIOUS DENSITY OF DBLP CO-AUTHOR DATASET.

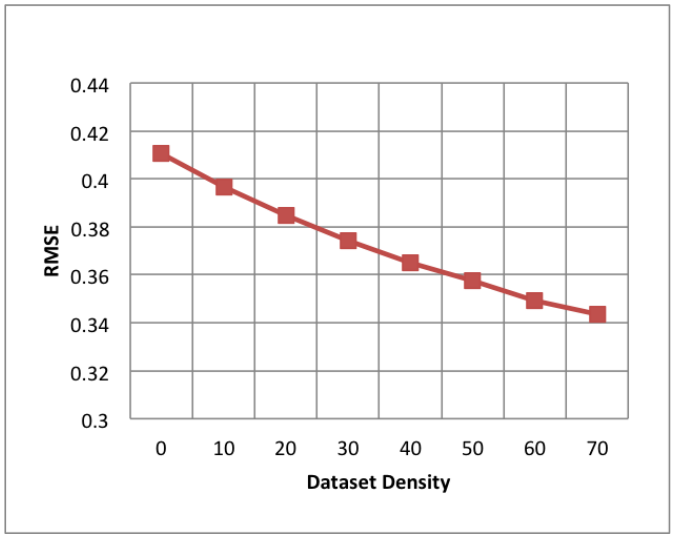

Figure 1. Performance of ALSMatrix-Factorization in terms of RMSE score with varying density of DBLP co-author co-author dataset

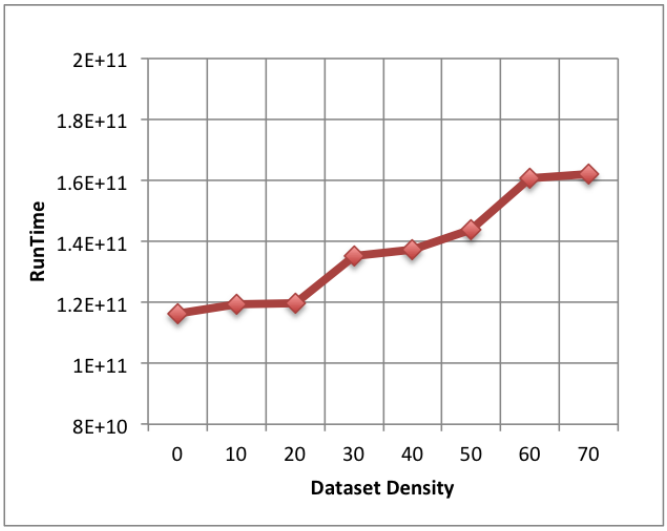

Figure 2. Performance of ALSMatrix-Factorization in terms of Run-time with varying density of DBLP co-author dataset.

the density of DBLP co-author dataset, the RMSE score which indicates the accuracy of the recommender model generated goes on improving.

Also as we increment the density of DBLP co-author dataset, we observe that run time complexity of ALS-Matrix-Factorization increases along with the accuracy of the recommended model. Accuracy is a prominently important feature of recommendation systems, but it is equally important that the model is computable under a reasonable amount of time; here the time constraint obviously depends on the application domain.As expected, in our case, the accuracy improves with the increasing density of DBLP dataset, which supports our hypothesis that the density of given feedback matrix determines the accuracy of the recommender model.

Discussion of Results: We conducted several experiments and evaluated the performance of Matrix Factorization on real-world implicit feedback datasets from DBLP co-author domain. ALS-Matrix-Factoriation in particular is able to achive good results as compared to the work presented in [20]. The increasing running time of the algorithm with the densification of dataset can be improved through parallelization and parameter tuning in general.

Also, analysis of our dataset revealed that for the DBLP dataset, there are many users in the who preferred only one or two different items and there are many items which are preferred by only one or two different users, popularly known as the Cold Start problem. One common way to address this problem when evaluating the performance of a recommendation algorithm is to filter out users who preferred too few items and to filter out items that are preferred by too few users [4], [18], [26]. However, given that we want to study the influence of dataset characteristics such as density of the datasets on the performance of the recommendation algorithms, we do not filter users or items with small number of preferences. This might be another reason for the average RMSE score obtained initially. But as we observed with the

increasing density which reduces cold start, the accuracy of the recommender model improved.

VI. Conclusion

In this work, we summarize our experiences and lessons learned from experiment with state-of-the-art collaborative filtering approach, Matrix Factorization, and implicit feedback dataset(DBLP co-author dataset). Overall, our study shows that knowledge about the characteristics of the domain and of specific data (density of dataset) can be used to guide an analyst towards the most appropriate algorithm to use, thus saving valuable time. The analysis of our results shows that for sparse data datasets (which are quite common in large-scale recommender systems), knowledge about the data domain (specifically, knowledge about the density of dataset)can be used to select the more suitable CF approach.

VII. FUTURE WORK

As part of future work, we would like to investigate the impact of change in parameters of MF on the accuracy of the recommender model generated. Since ALS is a general method for matrix factorization,as future work we intend to experiment with the proposed ALS variants on other application domains different from collaborative ltering. Also we would like to investigate ways to extend the existing CF algorithms for implicit feedback datasets to accommodate knowledge from multiple domains. We believe that many important networks in real- world can be modeled as heterogeneous interactions and the correlation between different networks can be used to improve recommendation accuracy.

References

[1] Netflix cinematch, September 2009.

[2] Dblp network dataset – KONECT, May 2015.

[3] Gediminas Adomavicius and Alexander 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, 2005.

[4] Shumeet Baluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, pages 895–904. ACM, 2008.

[5] 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*, pages 95–104. ACM, 2007.

[6] Robert M Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD-Cup and Workshop*, pages 7–14. ACM press, 2007.

[7] Robert M Bell, Yehuda Koren, and Chris Volinsky. The bellkor solution to the netflix prize, 2007.

[8] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[9] 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*, pages 271–280. ACM, 2007.

[10] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[11] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

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

[13] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[15] Michael Ley. The DBLP computer science bibliography: Evolution, research issues, perspectives. In *Proc. Int. Symposium on String Processing and Information Retrieval*, pages 1–10, 2002.

[16] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80, 2003.

[17] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*, 2014.

[18] Hao Ma, Irwin King, and Michael R Lyu. Mining web graphs for recommendations. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):1051–1064, 2012.

[19] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83, 1998.

[20] Rohit Parimi, Toma Trepka, Doina Caragea, and Cody Bennett. How to choose a recommender system: Insights and experiences for large-scale user personalization. In *Big Data (BigData Congress), 2015 IEEE International Congress on*, pages 475–482. IEEE, 2015.

[21] Huseyin Polat and Wenliang Du. Svd-based collaborative filtering with privacy. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 791–795. ACM, 2005.

[22] Francesco Ricci, Lior Rokach, and Bracha Shapira. *Introduction to recommender systems handbook*. Springer, 2011.

[23] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[24] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1. Citeseer, 2002.

[25] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456. ACM, 2011.

[26] Yang Yang, Niran Chawla, Yizhou Sun, and Jiawei Hani. Predicting links in multi-relational and heterogeneous networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 755–764. IEEE, 2012.

[27] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Algorithmic Aspects in Information and Management*, pages 337–348. Springer, 2008.