Movie Recommendation using Random Walks over the Contextual Graph

Toine Bogers
Information Interaction & Information Architecture
Royal School of Library and Information Science
Birketinget 6, DK-2300
Copenhagen, Denmark
tb@iva.dk

ABSTRACT

Recommender systems have become an essential tool in fighting information overload. However, the majority of recommendation algorithms focus only on using ratings information, while disregarding information about the context of the recommendation process. We present ContextWalk, a recommendation algorithm that makes it easy to include many different types of contextual information. It models the browsing process of a user on a movie database website by taking random walks over the contextual graph. We present our approach in this paper and highlight a number of future extensions with additional contextual information.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software

General Terms

Algorithms, Performance

Keywords

Recommender systems, context, random walks, contextual recommendation, movie recommendation

1. INTRODUCTION

Recommender systems are a form of personalized information filtering technology and have become an important tool for successfully dealing with the problem of information overload. Recommender systems have been applied to many different domains [10], with movie recommendation being an especially productive domain for recommendation technology. Some of the most popular data sets have come from the movie domain, such as the Movielens data set[1] and more recently the Netflix competition[2], and have resulted in a large body of work on movie recommendation. However, the majority of these and other approaches have focused exclusively on using ratings information for generating recommendations and improving the recommendation algorithms that use this type of information. While ratings are important, a user's enjoyment of a recommended movie is not exclusively dependent on how he[3] previously rated his movies.

What is missing from the majority of existing recommendation approaches is *context*. The context of the recommendation process can encompass a wide range of information, such as the time and location the recommendations are made, as well as the company the user requesting the recommendations is in. User context can be equally important to the recommendation process, such as mood, demographics, or social network information. Friendship relations between users have for instance been shown to be beneficial to recommendation performance [8, 12]. A third type of contextual information involves the context of the movies themselves, such as actor, director, and writer relations, or specific movie metadata, such as color and language.

So far, however, there have been few approaches that can incorporate such a variety of contextual information. Most recommendation algorithms have been designed to work with explicit ratings data or implicit usage data, due to the relative abundance of this type of information. In contrast, contextual information tends to be more difficult to collect, and is therefore often disregarded or left as future work. A second problem is that while it is easy to label certain information as context, it is much more difficult to produce a computable formalization of contextual information. This makes it difficult to explicitly model the great variety in contextual information.

In this paper, we present ContextWalk, a recommendation algorithm that can combine ratings information with contextual information for recommending movies. ContextWalk is based on taking Markov random walks over the contextual graph. In our algorithm, we model the browsing process of a user on a movie database website. We use the links between the different contextual objects on such websites, such as users, items, tags, genres, and actors, to construct a contextual graph. We then take a random walk along this graph, tempered by self-transitions to produce a

[1] http://www.grouplens.org/

[2] http://www.netflixprize.com/

[3] In the rest of this paper we use 'he' and 'his' to refer to both genders.

probability distribution over a user's unseen movies. The contextual graph could easily be expanded to include many more types of contextual information, making it easy to incorporate more context into the recommendation process.

In summary, we make the following three contributions:

- We propose ContextWalk, a recommendation algorithm based on Markov random walks that can combine ratings information with contextual information, such as tags, movie genre, actor, or director information to recommend movies.
- ContextWalk can easily be extended to include many additional contextual features, such as time, social network information, and mood information.
- ContextWalk can be used for many other recommendation tasks, such as tag recommendation or actor-to-movie recommendation, without the need for retraining or changing the recommendation model.

The remainder of this paper is structured as follows. In the next section, we explain our contextual recommendation model in more detail. Section 3 describes how our algorithm could be extended with additional contextual information. Section 4 discusses the most relevant related work. We conclude in Section 5.

2. CONTEXTUAL RANDOM WALKS

In our contextual recommendation model we model user behavior in a common scenario: browsing a movie database website such as the Internet Movie Database[4] (IMDB) or RottenTomatoes.com[5]. Movie database websites are a popular destination on the Web and browsing a movie database website can aid in exploration and discovery of new movies to watch. A user could, for instance, look up the page of a new movie he recently enjoyed and find more information about it. Perhaps he really enjoyed the performance of a specific actress in that movie, which could lead him to explore the actress' profile to find other interesting movies she might have starred in. Reviews, cast lists, and ratings could then guide the user to decide whether or not to see any of her other movies. Another possibility could be navigating to different movies based on one of the tags assigned to the original movie or the movie genre. Such a movie database website and the links between the different objects represented on them can be represented by a multipartite network (or *contextual graph*) such as the one shown in Figure 1.

The multipartite network in Figure 1 visualizes only a small number of different node types; such networks could easily be extended to included many other types of objects representing contextual information, such as writer, language, country of origin, production company, etc. However, in describing our contextual recommendation algorithm we will restrict ourselves to five types of contextual objects that are related to each other in some way on a movie database website: users (U), items[6] (I), tags (T), movie genres (G), and actors (A). Figure 1 shows a part of such a contextual

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

[5] http://www.rottentomatoes.com/

[6] In the context of this paper items are the same as movies. We use the term 'item' instead of 'movie' to keep in line with the accepted terminology in the field of recommender systems.

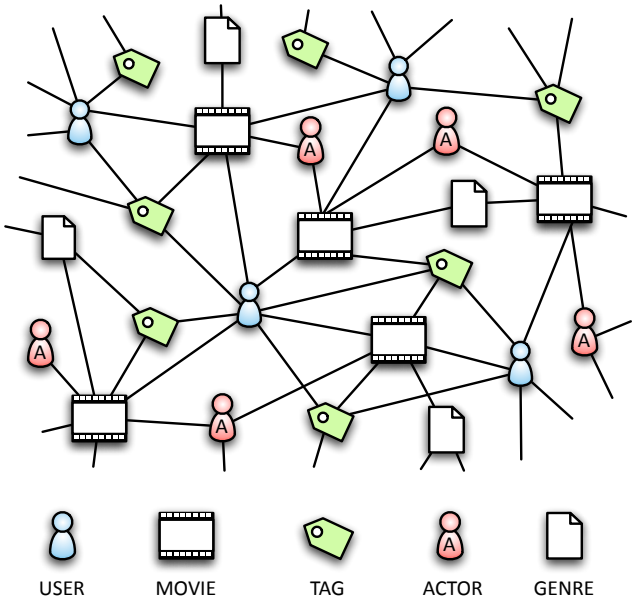

Figure 1: A subset of a contextual graph representing the content on a movie database website.

graph containing these five different node types and the links between them. Users can be connected to each other via movies or tags, and movies can share user, actors, genres, and tags. By following multiple links, a user could even browse to new movies that share the same actors, tags, or genres, even though the original movies were not connected directly.

2.1 Modeling browsing behavior

Our recommendation algorithm, ContextWalk, is based on modeling the browsing process of a user on a movie database website. In this browsing process, we assume that the user starts with a specific movie (or possibly another entity or contextual feature), and browses the contextual graph, until he finds an interesting node and stops the browsing process. If that node happens to be a movie, this could represent a user taking an interest in that movie with regard to future viewing. ContextWalk was inspired by the work by Craswell et al. [6], who successfully applied a random walk model to image search by modeling the query formulation process of users using the bipartite image-query graph. It was also heavily influenced by the work by Clements et al. [5], who used a random walk model for tag-based search on social bookmarking websites. We extend their models here to include contextual information for movie recommendation and emulate the user's browsing process by a random walk on the contextual graph.

Similar to [6], we make a number of simplifying assumptions in our model of user browsing behavior. The user has a limited memory, which means that he forgets his previous position on the contextual graph [6]. In contrast to algorithms like, for instance, PageRank [11] and FolkRank [9], we are not interested in the background probability of all nodes in the contextual graph by taking a random walk of infinite length. Instead, we limit our random walks to a finite length to keep the user in the vicinity of his original movie-related 'recommendation need' [6]. Our model is not

based on real browsing behavior on a movie database website, but instead estimates the transition probabilities from node to node using ratings and the links between different nodes, e.g., the actors associated with a movie. By using ratings instead of binary usage patterns, we can bias the random walk to start from the more likely movies, instead of assigning each movie the same starting point likelihood.

In our random walk model, we allow for the possibility of self-transitions, where the walk stays in place [6]. Self-transitions increase the influence of the initial state and provide another way of keeping the user in the vicinity of his original recommendation need. A self-transition corresponds to the user staying on the page of the currently selected movie, actor, tag, or genre, instead of moving on to another node. Another perspective on this model is viewing it as a noise process, where we start with a desired movie (as evident from the assigned ratings) and add noise by taking a number of steps. Given a starting movie, this produces in a probability distribution over all contextual nodes and corresponds to a Markov random walk [6].

2.2 Constructing the Contextual Graph

We explain our approach by focusing on contextual graph with five different types of nodes: users, items, tags, genres, and actors. Let $\mathcal{U}$ be the set of users, $\mathcal{I}$ be the set of items (or movies), $\mathcal{T}$ be the set of tags, $\mathcal{G}$ be the set of genres, and $\mathcal{A}$ be the set of actors[7]. We construct the contextual graph $G = (\mathcal{V}, \mathcal{E})$ as the union of these five sets $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{T} \cup \mathcal{G} \cup \mathcal{A}$. Let N be equal to $|\mathcal{V}|$, the number of nodes in G. The edges $\mathcal{E}$ in G correspond to the links between the different nodes, associating two nodes i and j with each other. For instance, for movie i and actor j we have an unweighted edge between i and j if actor j played in movie i. Edges between user and item nodes can either be represented by implicit binary usage patterns (seen or not) or by ratings. In this example, we assume that edges between user and item nodes are weighted by the ratings assigned to a movie by a user. By extending the graph in this way to include other contextual features, ContextWalk uses not only the links between users and items, as is common on collaborative filtering [13], but also links between tags, genres, and actor, all in the same model.

A random walk over G is a stochastic process in which the initial state is known and the next state S is governed by a probability distribution [5]. We can represent this distribution for our graph G by constructing the transition probability matrix $\mathbf{X}$, where the probability of going from node i (at time n) to node j (at time $n+1$) is represented by $\mathbf{X}_{i,j} = P(S_{n+1} = j | S_n = i)$.

Figure 2 shows how we create the transition probability matrix $\mathbf{X}$ for our contextual graph. It builds on the previous work done by [5]. The weights of the edges in our multipartite network are determined from the values in the individual sub-matrices. The **UI** sub-matrix contains ratings on a scale from 1 to 5, and the **UT** and **IT** sub-matrices contain the tag counts per user and per item respectively. All other sub-matrices are unary; each link between two object types is denoted by a 1. For example, each actor j in a particular movie i is represented in the item-actor matrix **IA** as $\mathbf{IA}_{i,j} = 1$. We combine the ten possible sub-matrices and their transpositions to produce $\mathbf{X}$ as shown in Figure 2. All

[7]Our notation borrows heavily from [5] and [6], which in turn borrowed from [14].

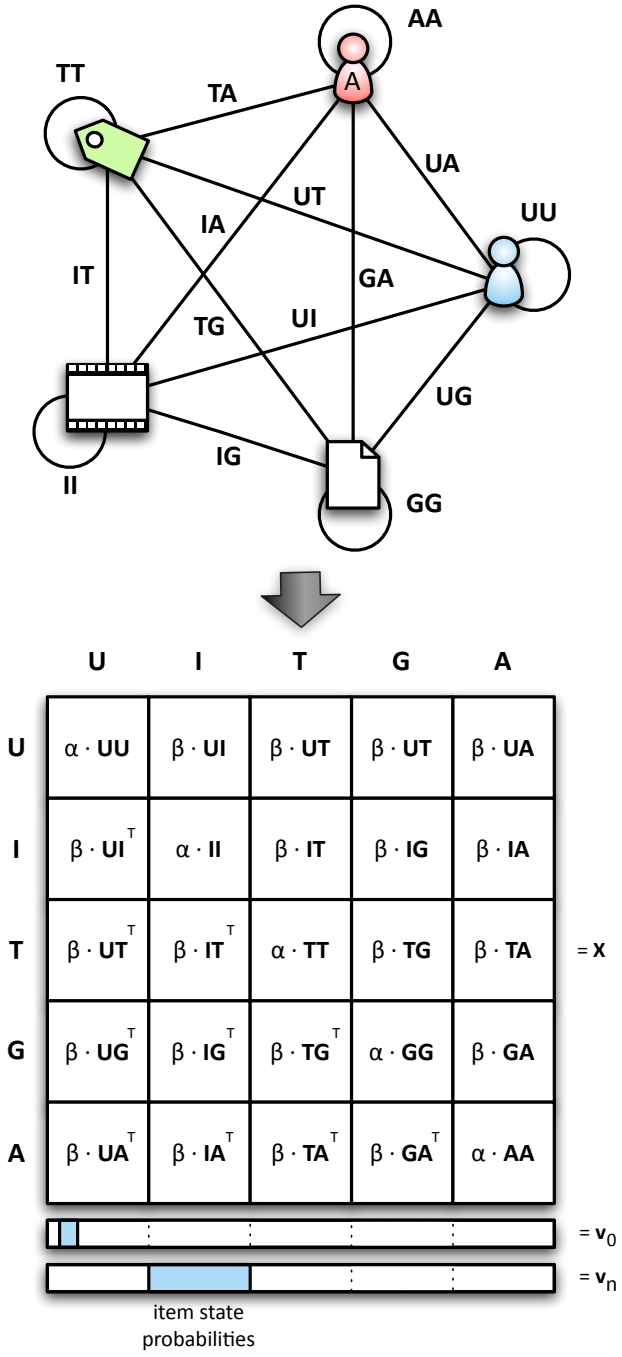

Figure 2: The weights of the edges in our multipartite network are determined from the values in the individual sub-matrices. The UI sub-matrix contains ratings on a scale from 1 to 5, and the UT and IT sub-matrices contain the tag counts per user and per item respectively; all other sub-matrices are unary; each link between two object types is denoted by a 1. Self-transitions in the contextual graph occur with probability α and are captured in the UU, II, TT, GG, and AA submatrices. We row-normalize X by multiplying the other sub-matrices by $\beta = \frac{1-\alpha}{\delta-1}$ where α is the self-transition probability and δ is the number of different contextual node types (i.e., the number of disjoint sets of nodes). This figure is adapted from the one in [5].

sub-matrices are row-normalized.

We represent the self-transitions that allow the walk to stay in place by adding the identity matrix to $\mathbf{X}$. They are captured in **UU**, **II**, **TT**, **GG**, and **AA** sub-matrices in Figure 2. These self-transitions are occur with probability α. To ensure that we can use the values in $\mathbf{X}$ as transition probabilities, we row-normalize $\mathbf{X}$ by a factor $\beta = \frac{1-\alpha}{\delta-1}$ where δ is the number of different contextual node types (i.e., the number of disjoint sets of nodes). This ensures that the row vectors of $\mathbf{X}$ sum to 1. For example, in the U-I-T-A-G case, δ is equal to 5.

Note that, while we have currently constructed a multipartite graph with five different types of nodes, we could also have restricted ourselves to, for instance, just users, items, and actors. The contextual graph for such a scenario and the accompanying transition probability matrix $\mathbf{X}$ would be constructed in the same manner as described above, but with smaller dimensions and with three sub-matrices instead of ten. The same construction principle holds for adding extra contextual node types to our graph, such as directors or writers.

2.3 Computing the Random Walk

To start our random walk over G we need to define an initial state vector $\mathbf{v}_0$, where we can select which user takes the walk by setting the element corresponding to that user to 1 and the other ones to 0. This is visualized in the bottom of Figure 2. We can then multiply the initial state vector $\mathbf{v}_0$ with the transition probability matrix $\mathbf{X}$ to calculate the transition probabilities $\mathbf{v}_1$ after taking one step on the contextual graph. In general, we can calculate multi-step probabilities either by multiplying the initial state vector $\mathbf{v}_0$ with the transitional probability matrix $\mathbf{X}^n$ after n steps or, equivalently and more efficiently, by iteratively multiplying the state vector for the previous step n by the transitional probability matrix: $\mathbf{v}_{n+1} = \mathbf{v}_n\mathbf{X}$. The state vector after any n steps contains the probability distribution over all nodes. As shown in the bottom of Figure 2, a part of $\mathbf{v}_n$ contains the item state probabilities after n steps on the contextual graph. After removing the items already rated by the user, we can then rank-order these probabilities to arrive at the recommendations for the user.

Note that our ContextWalk model also allows us to support different recommendation tasks. It is possible, for example, to support actor recommendation by extracting the actor state probabilities from $\mathbf{v}_n$, thereby recommending interesting new actors to the user. Another interesting application would be to 'activate' a set of actors in $\mathbf{v}_0$ instead of a user profile, and to generate movie recommendations based on these actors instead of a specific user profile. We could also easily support movie recommendation for a group of users instead of a single individual by 'activate' each of the group members in the initial state vector $\mathbf{v}_0$. By changing the weights of the individual members, we could even give certain users more influence on the recommendation process. Each of these tasks could be performed easily without having to alter or retrain the model.

3. CONTEXTUAL EXTENSIONS

In addition to the contextual features we currently include in our ContextWalk model, there are many different ways we could extend our model with other contextual information. In this section we list a few of the most promising examples.

Social networks Currently, ContextWalk does not utilize any information about similarity or social network relations between users. Instead, the only information represented in the **UU** sub-matrix is formed by the self-transitions. However, the **UU** sub-matrix could easily be extended to include information about for instance friendship relations between users in a social network instead of only self-transitions [8, 12].

Temporal information A common problem of many recommendation algorithms is that they do not include temporal information about recommendations. As a user's taste in movies might change, it would be essential for a recommender systems to assign a lower weight to movies that were watched a long time ago and higher weights to more recent movies, perhaps using some form of exponential weight decay function. While we have not included this in the algorithm presented in this paper, it would be relatively easy to include such temporal weighting. One solution would be to apply the weighting directly to the ratings matrix $\mathbf{R}$. Another, more elegant solution would be to alter the initial state vector $\mathbf{v}_0$ of a user. By adding the user's items to $\mathbf{v}_0$ with values between 0 and 1, weighted by age, a temporal context feature could easily be added to our ContextWalk model.

Item similarity Similarity between movies or other types of items could perhaps also be represented by other metrics such as textual similarity instead of only self-transitions in the **II** sub-matrix.

Tag similarity Similarly to item-item similarity, it is possible to determine the similarity between tags, perhaps using external resources such as WordNet [7].

Actor contribution Actors in a movie rarely have equal amounts of screen time. While it is currently unclear whether such a contextual feature would have a noticeable influence on movie recommendation, we could incorporate this into our model and test this. Instead of treating actor participation in a movie as a binary variable, we could instead weight their edges proportionately to the screen time they received.

4. RELATED WORK

Random walk models have been applied to recommendation several times before. Aggarwal et al. [2] were the first to apply a graph-theoretic approach to recommendation with their horting algorithm. Horting, as a type of random walk model, proved well-suited to dealing with the problem of sparsity. It can be extended with item metadata similarity to generated recommendations in the absence of user ratings. Hotho et al. [9] applied a random walk model to recommendation for social bookmarking. They construct a tripartite graph of users, items, and tags in a manner similar to ours, but without the self-transitions. Like PageRank [11], the FolkRank algorithm is based on a random walk model that calculates the fully converged state transition probabilities by taking a random walk of infinite length. ContextWalk was inspired partly by the work by Craswell et al. [6], who successfully applied a random walk model for

image search, by modeling the query formulation process of users using the bipartite image-query graph. It was also influenced by the work by Clements et al. [5], who propose a random walk model for tag-based item search. The difference between their approach and FolkRank is the inclusion of self-transition probabilities and using random walks of fixed length. Yildirim et al. [16] also present a recommendation algorithm that uses finite length random walks to produce item recommendations. They explicitly calculate item similarities using the cosine distance and use these similarities to infer the transition probabilities. Their model does not include any additional context beyond the user-item ratings. Yildirim et al. confirm the findings of [2] by showing that their random walk algorithm dealt well with sparsity. Wijaya et al. [15] also apply a random walk model to movie recommendation, but instead of constructing a graph based on usage patterns, they construct a sentiment graph of positive and negative terms in movie reviews. They calculate PageRank on the sentiment graph to rank the reviews and find that the ranking it computes is comparable to the ranking obtained from the box office figures. Finally, Baluja et al. [3] present a recommender system for YouTube videos based on random walks on the bipartite user-video graph. They evaluate their method on a three-month snapshot of live data from YouTube, and show that it outperforms using video co-views to recommend new videos.

5. CONCLUSIONS

In this paper we have presented ContextWalk, a movie recommendation algorithm based on taking random walks on the contextual graph. In addition to using the links between users and items as is common on collaborative filtering, it also allows for easy inclusion of many different types of contextual features, such as actors, genres, directors, writes, color, language, and so on. It also supports many other recommendation tasks with the same random walk model without the need for alteration or retraining, such as recommending interesting actors, recommending movies for a group of users, or tag recommendation.

5.1 Future Work

We are currently engaged in experiments with our ContextWalk model. Among other things, we wish to investigate the optimal combination of contextual features, i.e., whether a contextual graph with user, items, tags, genres, and actors outperform the graphs based on subsets on these five context types. We would also like to examine the benefits of extending our model as described in Section 3.

Acknowledgments

This research was supported by the Radio Culture and Auditory Resources Infrastructure Project (LARM) as funded by the Danish National Research Infrastructures Program (project no. 09-067292).

6. 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. *IEEE Transactions on Knowledge and Data Engineering*, 17(6): 734–749, 2005.

[2] C. C. Aggarwal, J. L. Wolf, K.-L. Wu, and P. S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In *Proceeding of KDD '99*, pages 201–212, New York, NY, USA, 1999.

[3] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video Suggestion and Discovery for Youtube: Taking Random Walks through the View Graph. In *Proceedings of WWW '08*, pages 895–904, New York, NY, USA, 2008.

[4] R. Burke. Integrating Knowledge-Based and Collaborative-Filtering Recommender Systems. In *Proceedings of the AAAI Workshop on AI in Electronic Commerce*, pages 69–72, 1999.

[5] M. Clements, A. P. de Vries, and M. J. Reinders. Optimizing Single Term Queries using a Personalized Markov Random Walk over the Social Graph. In *Proceedings of ESAIR '08*, 2008.

[6] N. Craswell and M. Szummer. Random Walks on the Click Graph. In *Proceedings of SIGIR '07*, pages 239–246, New York, NY, USA, 2007.

[7] C. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998.

[8] G. Groh and C. Ehmig. Recommendations in Taste-related Domains: Collaborative Filtering vs. Social Filtering. In *Proceedings of GROUP '07*, pages 127–136, New York, NY, USA, 2007.

[9] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. Information Retrieval in Folksonomies: Search and Ranking. In *Proceedings of ESWC '06*, 2006.

[10] M. Montaner, B. López, and J. L. de la Rosa. A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, 19(4):285–330, 2003.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[12] A. Said, E. W. De Luca, and S. Albayrak. How Social Relationships Affect User Similarities. In *Proceedings of the 2010 Workshop on Social Recommender Systems*, pages 1–4, 2010.

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *Proceedings of WWW '01*, pages 285–295, New York, NY, USA, 2001.

[14] M. Szummer and T. Jaakkola. Partially Labeled Classification with Markov Random Walks. *Advances in NIPS*, 2:945–952, 2002.

[15] D. T. Wijaya and S. Bressan. A Random Walk on the Red Carpet: Rating Movies with User Reviews and PageRank. In *Proceedings of CIKM '08*, pages 951–960, New York, NY, USA, 2008.

[16] H. Yildirim and M. S. Krishnamoorthy. A Random Walk Method for Alleviating the Sparsity Problem in Collaborative Filtering. In *Proceedings of RecSys '08*, pages 131–138, New York, NY, USA, 2008.