Web-Based Movie Recommender System

Chapter · January 2020

DOI: 10.1007/978-981-15-1518-7_24

5 authors, including:

Mala Saraswat
ABES Engineering College
27 PUBLICATIONS **84** CITATIONS
SEE PROFILE

Dr Anil Kumar Dubey
ABES Engineering College
65 PUBLICATIONS **113** CITATIONS
SEE PROFILE

Web-Based Movie Recommender System

Mala Saraswat, Anil Dubey, Satyam Naidu, Rohit Vashisht and Abhishek Singh

Abstract Recommender systems guide users to find and cull items such as restaurants, books, and movies from the huge collection of available options on the Web. Based on the user's taste and preferences, recommender system recommends a set of items from a large set of available items. Recommendation systems include intelligent aides for filtering and selecting Web sites, news stories, TV listings, and alternative info. The users of such systems typically have various conflicting desires, variations in personal preferences, social and academic backgrounds, and personal or skilled interests. As a result, it looks fascinating to possess personalized intelligent systems that suit individual preferences. The need for personalization has proliferated the event of systems that adapt themselves by ever-changing their behavior supported by the inferred characteristics of the user interacting with them. In this paper, we develop a Web-based movie recommender system that recommends movies to users based on their profile using different recommendation algorithms. We also compare various recommendation algorithms such as singular value decomposition, alternating least squares and restricted Boltzmann machines.

Keywords Recommender systems · Collaborative filtering · KNN · Singular value decomposition · Alternating least squares and restricted Boltzmann machines

1 Introduction

"Enhancing the performance of recommender systems" deals with improving performance of recommender systems applicable to various domains [1]. Netflix recommends TV serials and movies based on what you have watched and what other Netflix users with the same interest have watched. Amazon also recommends a product item to a user based on what other customers who purchased that item in which

M. Saraswat (✉) · A. Dubey · S. Naidu · R. Vashisht · A. Singh
Department of Computer Science and Engineering, ABES Engineering College, Ghaziabad, India

A. Dubey
e-mail: anil.dubey@abes.ac.in

Y.-C. Hu et al. (eds.), *Ambient Communications and Computer Systems*, Advances in Intelligent Systems and Computing 1097,
https://doi.org/10.1007/978-981-15-1518-7_24

a user might be interested. Broadly recommender systems are categorized into collaborative filtering (CF) and content-based filtering (CB). CF identifies similarity among users based on their past behavior and then recommends items to the user which are liked, bought, or rated highly by similar users [2]. This recommender system can predict items to the user might have an interest, even though the user has never expressed explicit interest. This is generally called user-user collaborative filtering. The opposite of user-user collaborative filtering is to find items similar to a given item and recommend items to users who have also liked, bought, or rated other similar items highly. This goes by the name item-item collaborative filtering.

Content-Based Filtering relies on features of the items based on their content [3]. Based on users' ratings on existing items, a user profile is created and the ranks provided by the users are given to those items. Based on user liking for items, content-based filtering learns the distinct properties of those items and recommend additional items with similar properties. In content-based filtering, the user is recommended items based on their preferences. This does not involve how other users have rated the items.

Recommender systems deal with certain limitations such as *cold start*, *data scalability* and *sparsity*. Cold-start problem is concerned with the issue that the recommender system cannot draw any inferences for users or items who are new to the system and have not rated the items. There are basically two types of problem. The first one is a new user cold-start problem where a new user without having any ratings to show his/her taste enters the system. Second is new item cold-start problem when a new item which has not been sufficiently been rated enters the system. The next problem is scalable data. With advent of Web 2.0, there is proliferation of e-commerce sites. People sell hundreds of millions of products to millions of users online. So new algorithms and scalable systems should be designed to produce high-quality recommendations even for very large-scale problems. The third problem is data sparsity. Users who are very active contribute to the rating for a few number of items available in the database of movies or books items. Besides some popular items are only rated. Because of rating sparsity, the similarity between two users or items cannot be defined, rendering collaborative filtering useless.

Recommendation system is a software tool that provides suggestions for items to a user [4]. In this work, we design a Web interface and compare various recommendation algorithms for movie domain. Our Web-based recommendation system is a component of a larger system which is a music streaming and downloading Web application. The application integrates our system to main Web application as a Web service. The service provides the opportunity of displaying latest movies and user lists along with accurate information. Moreover, there are two types of lists, professional lists provided by application and user lists created by individual user. Users can display both the types of lists. All these actions generate dataset to our system for accurate recommendations based on users' and items' similarities. The dataset is the most important part of our system; therefore, main system's existence and wide usage are important for generating recommendations on our system.

2 Different Approaches for Recommendation

2.1 K-Nearest Neighbor (KNN)

KNN algorithm for collaborative filtering finds k most similar neighbors of users or items in the feature space based on the similarity of ratings. Based on the similarity of ratings users' are recommended items similar to those as those liked by similar users. K-nearest neighbor finds the k most similar items to a particular instance based on a given distance metric like Euclidean, Jaccard similarity, Minkowsky, etc. [5, 6].

2.2 Singular Value Decomposition (SVD)

Besides CF and CF-based recommender systems, latent factor-based filtering recommendation methods attempt to discover latent features to represent user and item profiles by decomposing the ratings [7]. Unlike the content-based filtering features, these latent features are not interpretable and can represent complicated features. For instance, in a movie recommendation system, one of the latent features might represent a linear combination of humor, suspense, and romance in a specific proportion.

Generally, for already rated items, the rating r_{ij} given by a user i to an item j can be represented as:

$$r_{ij} = u_i^T v_j \tag{1}$$

where u_i is the user profile vector based on the latent factors and v_i is the item vector based on the same latent factors

$$R = USV^T = US \tag{2}$$

One of the ways these user and item profiles can be created is by performing singular value decomposition (SVD) on the rating matrix after filling in the missing values by some form of mean values across the users and items as appropriate. According to SVD, the rating matrix R can be decomposed as follows:

$$R = USV^T = US^{\frac{1}{2}} S^{\frac{1}{2}} V^T \tag{3}$$

We can take the user profile matrix as $US^{\frac{1}{2}}$ and then transpose of the item profile matrix as $S^{\frac{1}{2}} V^T$ to form the latent factor model.

2.3 Alternating Least Squares (ALS)

ALS is an iterative optimization process that builds a matrix factorization CF-based model. For every iteration, we try to arrive closer and closer to a factorized representation of our original data [8].

In ALS, we have a rating matrix R of size $u \times i$ where u is the number of users, i is the number of item. The rating matrix is factored into one matrix with users and hidden features U of size $u \times f$ and one V with items and hidden features of size $f \times i$. U and V matrices have weights corresponding to how each user/item relates to each feature. We find values of U and V so that their product approximates R as closely as possible as shown in Eq. 4:

$$R \approx U \times V. \tag{4}$$

2.4 Restricted Boltzmann Machines (RBM)

RBMs have two layers: input layer or **visible** layer and outer or the **hidden** layer. The neurons in each layer communicate with neurons in the other layer but not with neurons in the same layer. Thus, there is no intra-layer communication among the neurons [9] as shown in Fig. 1.

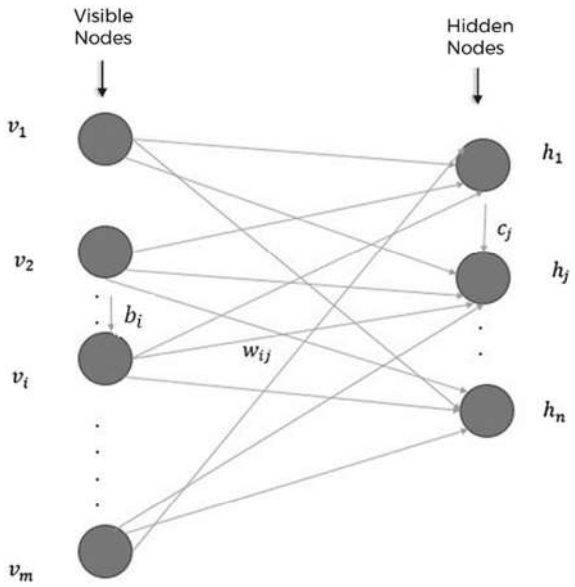

Fig. 1 Restricted Boltzmann machines

Restricted Boltzmann machines can be used to build a recommender system for items ratings. The RBM recommender system can learn the probability distribution of ratings of items for users given their previous ratings and the ratings of users to which they were most similar to. Then RBM recommender system used the learned probability distribution to predict ratings on never-before-seen items [10].

- The hidden layer is used to learn features from the information fed through the input layer.
- The input is going to contain X neurons, where X is the amount of movies in our dataset.
- Each of these neurons will possess a normalized rating value varying from 0 to 1: 0 meaning that a user has not watched that movie and the closer the value is to 1, the more the user likes the movie that neurons representing.
- These normalized values will be extracted and normalized from the ratings dataset.
- After passing in the input, we train the RBM on it and have the hidden layer learn its features.
- These features are used to reconstruct the input, which will predict the ratings for movies that the input has not watched, which is what we can use to recommend movies.

3 Experiments and Results

The Web application has a user-friendly graphical user interface (GUI) to provide ease of use and effectiveness to the users with different roles. Our Web service will be integrated to Web application and the recommendation result will be displayed through Web application GUI. In the Web service interface, there are four pages Login, Lists, Recommendations and Profile. This Login page contains a login text fields and a few buttons. The users of the system who want to get recommendation fill the text field with proper user id. After he fills the text field and click the submit button, new page is opened and recommendations are shown in this page. In Lists page, there will be different sections according to the number of lists created by the user. The Profile page will show user information like name, user id, movie ratings, etc. Figure 2 shows the use case diagram. Figures 3, 4 and 5 depict data flow diagram level 0–2. Recommendation System is composed of the following fundamental features:

(1) Users:
    (a) Generate Data (b) Get Recommendation
(2) Inter-agent:
    (a) Get Recommendation (b) Provide Dataset (c) Update Dataset (d) Integrate Web Service Experiments are conducted on the data that user fill based on their liking in the web interface (Fig. 6).

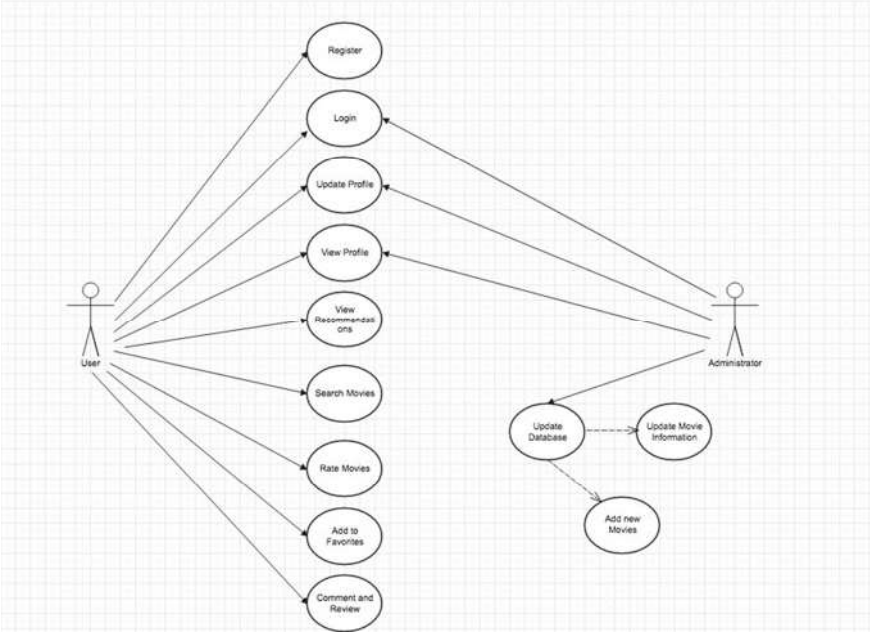

Fig. 2 Use case diagram

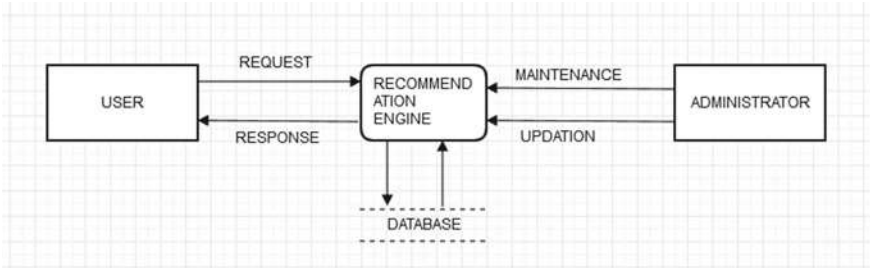

Fig. 3 Data flow diagram level 0

3.1 Evaluation Metrics

The goal of recommender system is not to recommend the same product that the user used/liked before. So to evaluate your recommendation model some metrics are designed such as precision, recall, F-measure, Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) [1]. For our models, we have used RMSE and MAE. They are used to evaluate accuracy of a filtering technique by comparing the predicted ratings directly with the actual user rating:

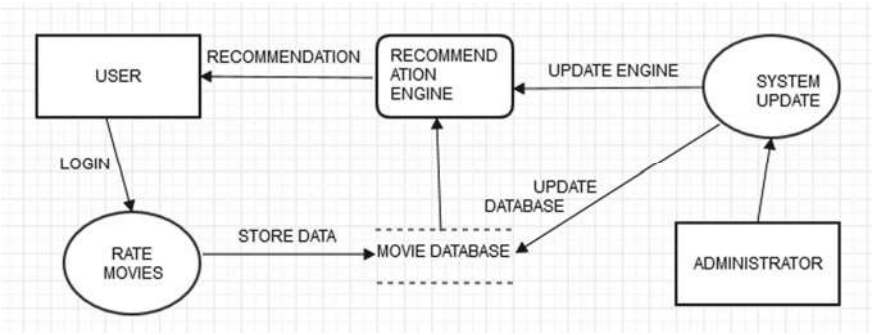

Fig. 4 Data flow diagram level 1

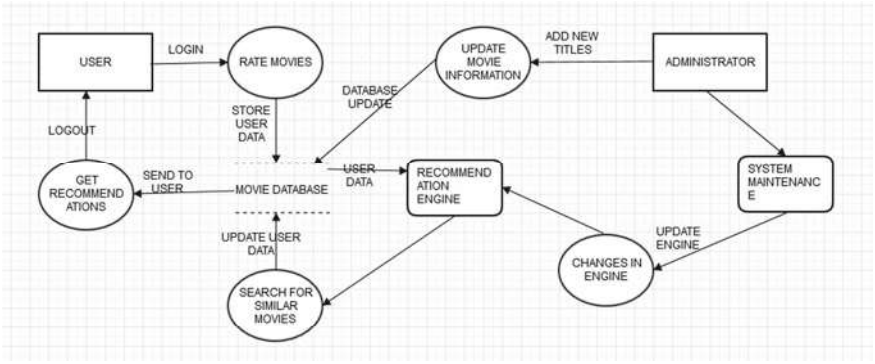

Fig. 5 Data flow diagram level 2

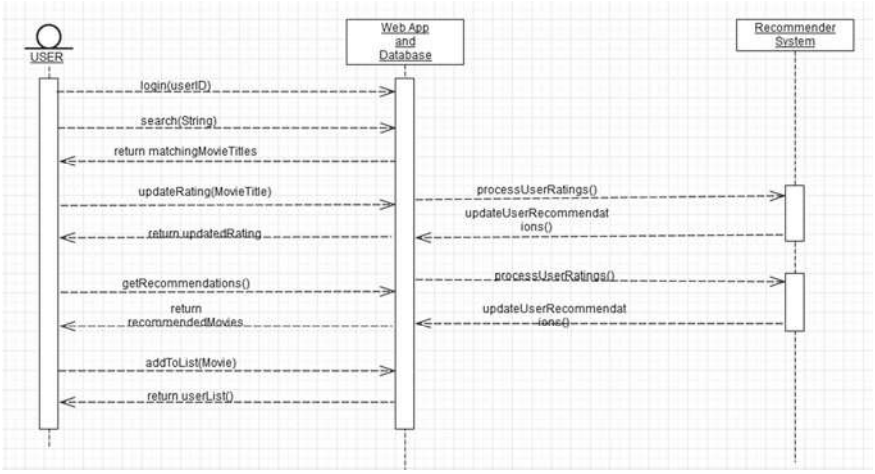

Fig. 6 Sequence diagram

(1) Mean Absolute Error: MAE is a measure of deviation of predicted rating or recommendation from user's actual rating

$$\text{MAE} = \frac{1}{N} \sum |\text{predicted rating} - \text{actual rating}| \tag{5}$$

(2) Root Mean Square Error: RMSE is the standard deviation of the prediction errors. RMSE provides higher weights to large errors compared to trivial errors. Thus, RMSE is more used when large errors are particularly undesirable.

$$\text{RMSE} = \sqrt{\sum \text{predicted} - \text{actual})^2} \tag{6}$$

Recommendation engine predicts more accurately user ratings with lower MAE and RMSE values. RMSE and MAE evaluate how accurate our prediction ratings are compared to actual rating and thus enable us to find out the quality of our recommendations.

3.2 Comparison of Various Recommender System Algorithms

We have developed a Web-based recommender system that recommends movies using various algorithms. We then compare these algorithms using performance metrics. Figure 7 shows the snapshot of Web interface of our Web-based recommender system through which user login to get recommendations.

Figure 8 shows the snapshot of various movies recommended to the users based on his interest. Table 1 summarizes the performance of various recommender system algorithms using RMSE and MAE as discussed in Sect. 3.1. For both user- and item-based K-nearest neighbor approach using collaborative filtering, KNN produces a 0.9375 RMSE and 0.7263 MAE which is good but the recommendations produced are not up to mark. Single Value Decomposition (SVD) has RMSE score of 0.9002 and MAE score of 0.6925 whereas ALS produces RMSE of 1.069 and MAE 0.803. RBM produces highest RMSE of 1.1897 and MAE of 0.9935 with worst performance (Fig. 9).

4 Conclusions and Future Directions

From experiments, algorithm based on SVD provides better recommendation as compared to KNN, ALS and RBM. The restricted Boltzmann model (RBM) of Deep Learning produces better recommendations when compared to several content-based and Collaborative-Based algorithms but when compared on the basis of RMSE (Root Mean Square Error), MAE (Mean Absolute Error), we see that the performance is

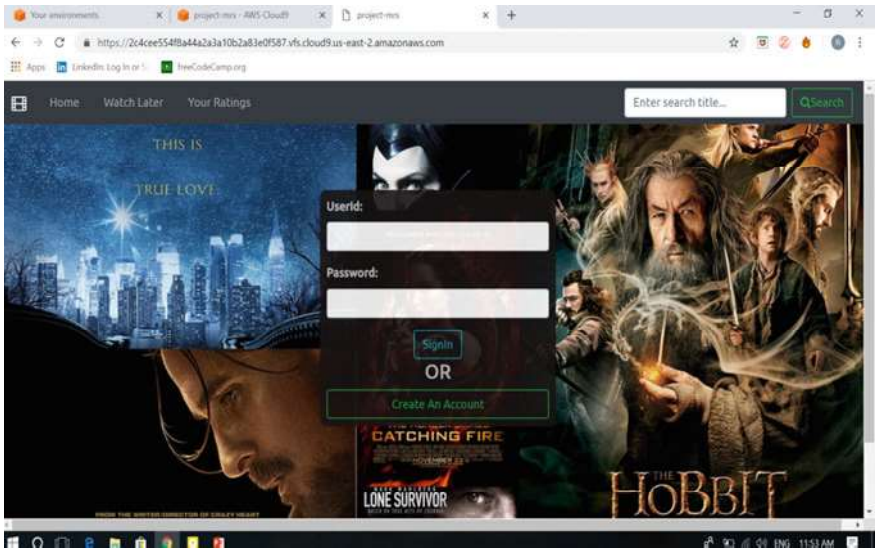

Fig. 7 Snapshot showing user login Web interface

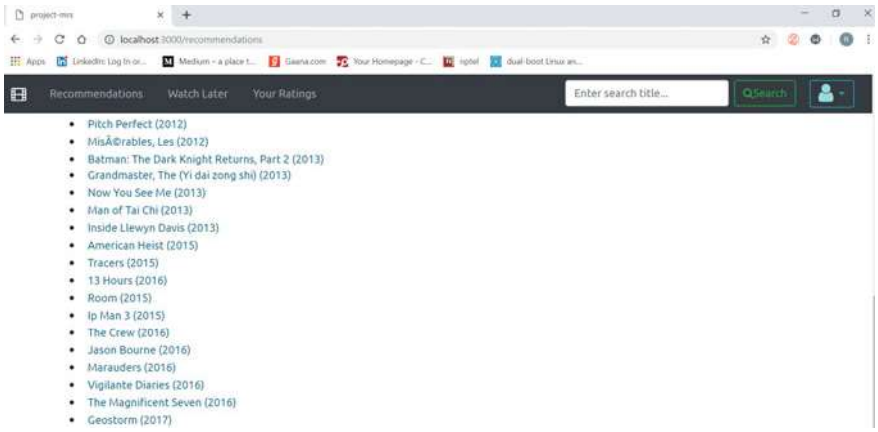

Fig. 8 Snapshot showing various movie recommendations to the user

Table 1 Performance of various recommendation algorithm

	RMSE	MAE
SVD	0.9002	0.6925
ALS	1.0687	0.803
KNN	0.9375	0.7263
RBM	1.1897	0.9935

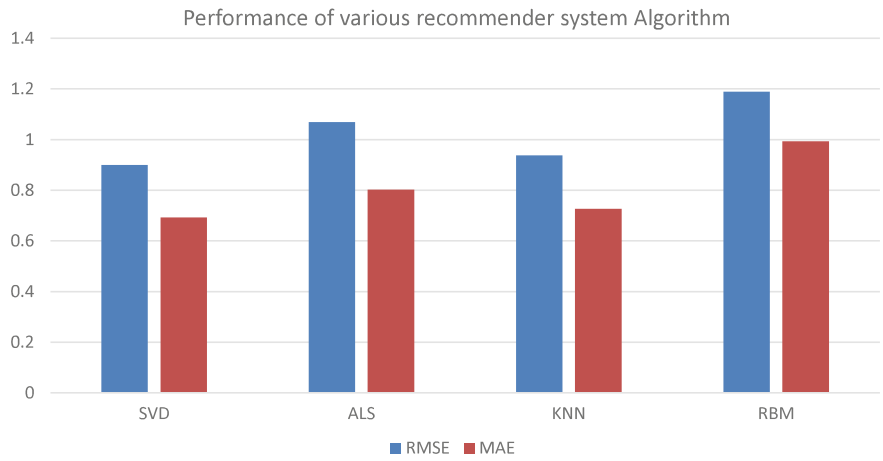

Fig. 9 Chart showing performance of various recommender system algorithm

not good. This is because user needs to enter sufficient ratings to learn the model. Users need to input their ratings to produce good quality recommendations.

Future direction is to build deep factorization machines and is to combine the power of factorization machines with the power of deep neural networks to create even more powerful and improved recommendations. If we somehow manage to mix these two powerful features together, we could unlock the ways to achieve commendable recommendations which could fulfill everyone's requirements.

References

1. Adomavicius, Gediminas, and Alexander Tuzhilin. 2005. 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* 6: 734–749.
2. Pazzani, Michael J., and Daniel Billsus. 2007. Content-based recommendation systems. *The adaptive web*, 325–341. Berlin, Heidelberg: Springer.
3. Schafer, J.B., D. Frankowski, J. Herlocker, and S. Sen. 2007. Collaborative filtering recommender systems. *The adaptive web*, 291–324. Berlin, Heidelberg: Springer.
4. Resnick, P., and H.R. Varian. 1997. Recommender systems. *Communications of the ACM* 40 (3): 56–58.
5. Linden, G., B. Smith, and J. York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing* 7 (1): 76–80.
6. Konstan, J., B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl. 1997. GroupLens: Applying collaborative filtering to Usenet news. *Communications of the ACM* 40 (3): 77–87.
7. Sarwar, B.M., G. Karypis, J.A. Konstan, and J. Riedl. 2000. Application of dimensionality reduction in recommender system—A case study. In *ACM WebKDD'00 (Web-mining for ECommerce Workshop)*.
8. https://datasciencemadesimpler.wordpress.com/tag/alternating-least-squares/.

9. Hinton, G.E., and R.R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313 (5786): 504–507.
10. Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning*. ACM.