From Matrix Factorization To Deep Neural Networks: The Evolution Of Collaborative Filtering Solutions for Recommendation Systems

James Le
Rochester Institute of Technology
jl1165@rit.edu

Alexander G. Ororbia
Rochester Institute of Technology
ago@cs.rit.edu

ABSTRACT

Recommendation systems are technologies and techniques that can provide recommendations for items to be of use to a user. The recommendations provided are aimed at supporting their users in various decision-making processes, such as what products to purchase, what music to listen, or what routes to take. Correspondingly, various techniques for recommendation generation have been proposed and deployed in commercial environments. The goal of this independent study is to impose a degree of order upon this diversity by presenting a coherent and unified repository of the most common recommendation methods to solve the collaborative filtering problem: from classic matrix factorization to cutting-edge deep neural networks.

KEYWORDS

recommendation systems, deep learning, collaborative filtering, matrix factorization, neural networks

1 INTRODUCTION

Recommendation systems are built to predict what users might like, especially when there are lots of choices available. They can explicitly offer those recommendations to users (e.g., Amazon or Netflix, the classic examples), or they might work behind the scenes to choose which content to surface without giving the user a choice.

Either way, these systems are critical for certain types of businesses because they can expose a user to content they may not have otherwise found or keep a user engaged for longer than they otherwise would have been. While building a simple recommendation system can be quite straightforward, the real challenge is to actually build one that works and where the business sees real uplift and value from its output.

Recommendation systems can be built using a variety of techniques, from simple (e.g., based only on other rated items from the same user) to extremely complex. Complex recommendation systems leverage a variety of different data sources (one challenge is using unstructured data, especially images, as the input) and machine learning (including deep learning) techniques. Thus, they are well suited for the world of artificial intelligence and more specifically unsupervised learning; as users continue to consume content and provide more data, these systems can be built to provide better and better recommendations.

1.1 Recommendation Systems Function

There are a variety of reasons as to why recommendation systems can be deployed in the business world. The 4 reasons below, in my opinion, stand out:

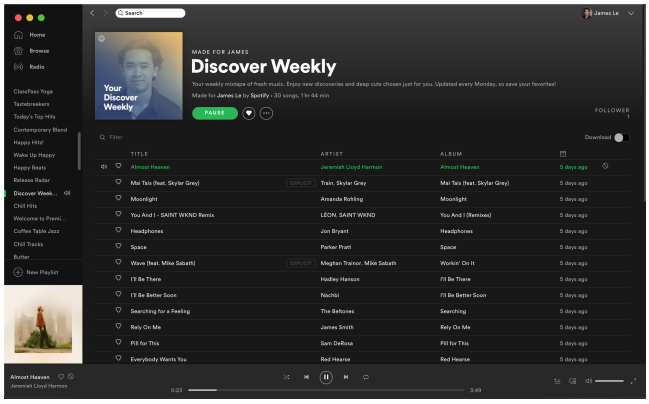

Figure 1: Spotify Discover Weekly

- *To increase the number of items sold*: In general, from a business perspective, the primary goal for using recommendation systems is to increase the conversion rate - which is the number of users that accept the recommendation and consume an item, compared to the number of visitors that just browse through the information.
- *To sell more diverse items*: In a music streaming service such as Spotify, the business is interested in displaying all the music from all types of artists, not just the popular ones. This could be difficult to achieve without a recommendation system, since Spotify cannot afford the risk of advertising music that are not likely to suit a particular user's taste. Therefore, a recommendation system can suggest unpopular music to the right users (Figure 1).
- *To increase user satisfaction*: A well designed recommendation system can improve the user experience within the application. The user will find the recommendations to be interesting, relevant and enjoyable, thus leading to higher satisfaction.
- *To better capture user's needs*: A good recommendation system can describe the user's preferences in detail, either collected explicitly or implicitly. The business may then decide to reuse this knowledge for a number of other goals.

1.2 Application and Evaluation

Recommendation system research is being conducted with a strong emphasis on practical and commercial applications. In fact, the application domain has a major effect on the algorithmic approach

that should be taken. [36] provides a taxonomy of recommendation systems and classifies their existing applications to specific domains:

- *Entertainment* - recommendations for movies (Netflix), music (Spotify, Pandora), and mobile apps (Apple Store, Android Store).
- *Content* - personalized newspapers (The New York Times, The Wall Street Journal), recommendation for images (Pinterest), recommendations of Web pages (Pocket), e-learning applications (Coursera), and e-mail filters (Gmail).
- *E-commerce* - recommendations for consumers of products to buy such as books (Amazon), beauty supplies (Birchbox), and clothes (Stitch Fix).
- *Services* - recommendations of travel services (Skyscanner), experts for consultation (StyleSeat, ClassPass), houses to rent (Zillow), or matchmaking services (Tinder, Bumble).

1.3 Existing Challenges

There are many challenging topics that are important for research work on recommendation systems. The 5 below captured my attention:

- *Scalability*: A fundamental issue for recommendation systems is how to deploy the algorithms in production and handle massive and dynamic real-world datasets, which are generated by the interactions of users with items (such as ratings, preferences, reviews etc.) A solution that works well with small offline datasets can't be guaranteed to perform well on large online datasets. Many studies on large-scale evaluation have been proposed to focus on this issue ([16], [21], [38], [46])
- *Proactivity*: In the scenarios emerging today, where computers are ubiquitous and users are always connected, it seems natural to imagine that a recommendation system can detect implicit requests. It therefore needs to predict not only what to recommend, but also when and how to provide its recommendations proactively ([44], [30], [34]).
- *Privacy*: In the attempt to build increasingly better and more personalized recommendations, recommendation systems collect as much user data as possible. This will clearly have a negative impact on the privacy of the users and therefore, there is a need to design solutions that will parsimoniously and sensibly use user data ([42], [11], [40], [27], [8], [49], [2], [54], [33]).
- *Diversity*: In the early stage of a recommendation process, the users are likely to explore new and diverse directions. In such cases, the recommendation system can be utilized as a knowledge discovery tool. Many research studies ([51], [31], [32]) have attempted to characterize the nature of diversity - whether diversity among different recommendation sessions or within a session, and how to combine diversity and accuracy goals together.
- *Generalizability*: Generic user models and cross domain recommendation systems are able to mediate user data through different systems and application domains ([6], [25], [7], [24]). Using generic user model techniques, a single recommendation system can produce recommendations about a variety of items. You can think of it to be similar to "transfer learning" in deep neural networks.

2 PROBLEM FORMULATION

Mathematically speaking, the recommendation problem can be formulated as follows: Let U be the set of all users and let I be the set of all items. Both of these spaces can be very large - millions in many consumer services. Let f be the utility function that measures the usefulness of item i to user u, as such: $f : U \times I \rightarrow R$, where R is a totally ordered set. Then, for each user $u \in U$, we want to choose an item $i' \in I$ that maximizes the user's utility. Mathematically:

$$\forall u \in U, i'_s = \arg\max_{i \in I} f(u, i)$$

In recommendation systems, the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item. In general, utility can be an arbitrary function, including a profit function. Depending on the application, the utility f can either be specified by the user, as is often done for the user-defined ratings, or is computed by the application, as can be the case for a profit-based utility function.

Each element of the user space U can be defined with a profile that includes various user characteristics, such as user ID, age, gender, income etc. Similarly, each element of the item space I is defined with a set of characteristics. For example, in a music recommendation application, where I is a collection of songs, each song can be represented not only by its ID but also by its title, genre, artist, year of release etc.

The central problem of recommendation system is that the utility f is usually not defined on the whole $U \times I$ space, but only on a subset of it. This means f needs to be extrapolated to the whole space $U \times I$. In recommendation systems, utility is typically represented ratings and is initially defined only on the items previously rated by the users. For example, in a movie recommendation application, users initially rate some subsets of movies that they have already seen. The goal of the application is to predict the ratings of the non-rated movie/user combinations and display appropriate recommendations based on such predictions.

Extrapolations from known to unknown ratings are usually done by either: (1) specifying the heuristics that define the utility function and empirically validating its performance; or (2) estimating the utility function that optimizes certain performance criteria, such as root mean squared error.

Once we are able to estimate the unknown ratings, we can make the actual recommendations of an item to a user by choosing the highest rating among all the estimated ratings for that user. Otherwise, we can also recommend the K best items to a user or a set of K users to an item.

Generally speaking, recommendation systems are classified into 3 categories [3]: (Figure 2)

- *Content-Based Approaches*: The user will be recommended items that are similar to the ones that he/she preferred in the past.
- *Collaborative Filtering Approaches*: The user will be recommended items that people with similar tastes and preferences liked in the past.

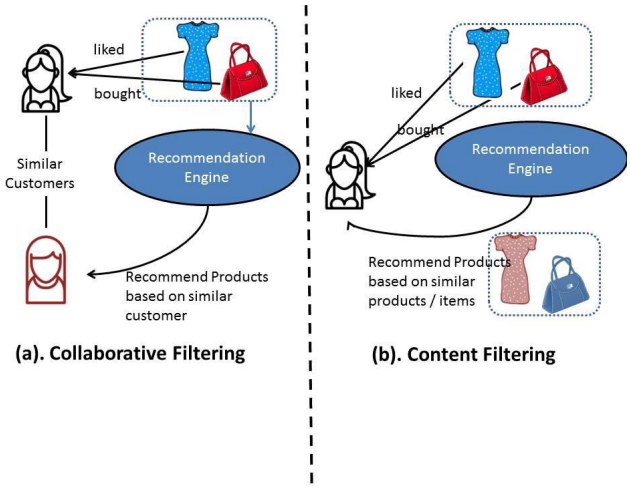

Figure 2: Content-Based Filtering vs Collaborative Filtering

- *Hybrid Approaches*: These combine the previous two approaches.

2.1 Content-Based Approaches

In content-based recommendation systems, the utility $f(u, i)$ of item i for user u is estimated based on the utilities $f(u, i_k)$ assigned by user u to item $i_k \in I$ that are similar to item i. For instance, in a music recommendation engine, in order to recommend new songs to user u, the content-based recommendation system tries to understand the similarities among the songs that user u has listened frequently in the past. Then, only the songs that have a high degree of similarity to whatever the user's preferences are would be recommended.

As was observed in [3] and [48], content-based approaches have several limitations:

- *Limited Analysis*: Content-based approaches are limited by the features that are explicitly associated with the items recommended. Hence, in order to have a sufficient set of features, the content must either be in a form that can be processed automatically or that can be assigned to items manually. In the first scenario, automatic feature extraction methods are much harder to apply to multi-media data such as images and audio. In the second scenario, it is often impractical to assign attributes by hand because of limited computational resources.
- *Homogeneity*: When the system can only recommend items that score highly against a user's profile, the user is restricted to only items that are similar to those already rated. In other words, content-based approaches do not have much weight on the diversity of the recommendations. Ideally, the user should be presented with a range of options and not with just a homogeneous set of alternatives.
- *Cold Start Problem*: The user has to rate a sufficient number of items before a content-based recommendation system can understand the user's preferences and present the user with trustworthy items. Thus, a new user with no previous ratings would not be able to get accurate recommendations.

2.2 Collaborative Filtering Approaches

In collaborative filtering recommendation systems, we attempt to predict the utility of items for a particular user based on the items previously rated by other users. More formally, the utility function $f(u, i)$ of item i for user u is estimated based on the utilities $f(u_j, i)$ assigned to item i by those users $u_j \in U$ who are similar to user u. For instance, in a book recommendation application, in order to recommend books to user u, the collaborative filtering system finds the peers of user u, other users with similar tastes in books (rate the same books similarly). Then, only the books that are most liked by the peers of user u would be recommended.

According to [9], algorithms for collaborative filtering approaches can be divided into two classes: *memory-based* and *model-based*. Memory-based algorithms are heuristics that make rating predictions based on the entire collection of previously rated items by the users. In contrast to memory-based methods, model-based algorithms use the collection of ratings to learn a model from the underlying data using statistical and machine learning techniques, which is then used to make rating predictions. A method combining both approaches was proposed in [5], where it was empirically demonstrated that the use of this combined approach can provide better recommendations than pure memory-based and pure model-based techniques.

The pure collaborative recommendation systems do not have the shortcomings that content-based systems have. However, they still have their own limitations:

- *New User Problem*: This is the same problem as with content-based approaches. In order to make accurate recommendations, the system must first learn the user's preferences from the previous ratings.
- *New Item Problem*: New items are regularly inputted to the system. Because collaborative filtering approaches rely solely on users' preferences to make recommendations; until the new item is rated by a sufficient number of users, it won't be able to get recommended.
- *Sparsity*: In any system, the number of ratings already obtained is usually small compared to the number of ratings that need to be predicted. Effective prediction of ratings from a small number of examples is quite important. Also, the success of collaborative filtering depends on the availability of a critical mass of users. For example, in the product recommendation system, there may be many products that have been purchased by only few people and they would be recommended very rarely, even if those few users gave high ratings to them. Furthermore, for the user whose tastes are unusual compared to the rest of the population, there will not be any other users who are particularly similar, leading to poor recommendations [3].

2.3 Hybrid Approaches

Several recommendation systems use a hybrid approach by combining content-based and collaborative filtering methods, which

helps to avoid certain limitations of them. Different ways to combine content-based and collaborative methods can be classified as follows:

- Implementing collaborative and content-based methods separately, then combining their predictions. ([13], [39])
- Incorporating some content-based characteristics into a collaborative filtering recommendation system. ([3], [39], [18], [35])
- Incorporating some collaborative filtering characteristics into a content-based recommendation system [50].
- Constructing an unifying model that incorporates both content-based and collaborative filtering characteristics. ([4], [41], [47], [1], [10])

2.4 Deep Learning Approaches

Recently, deep neural networks have been revolutionizing the recommendation architectures dramatically and brings more opportunities to improve the performance of recommendation systems. Recent advances in deep learning based recommendation systems have gained significant attention by overcoming obstacles of conventional models and achieving high recommendation quality.

One of the most attractive properties of deep neural network architectures is that they are (1) end-to-end differentiable and (2) provide suitable inductive biases personalized to the input data type. Therefore, if there is an inherent structure that the model can exploit, then deep neural networks can be very useful.

When dealing with content-based recommendation, the key advantage is that in deep neural networks, multiple neural layers can be stacked into a single differentiable function and trained end-to-end. For instance, when dealing with text and image data, Convolutional Neural Networks and Recurrent Neural Networks [19] become indispensable neural building blocks. The traditional approach is less attractive because it cannot take advantage of end-to-end representation learning, while the deep learning framework can do son in a unified joint framework ([59]).

When dealing with collaborative filtering recommendation, the key advantage is that deep neural networks can handle huge amount of complexity and large number of training data. Traditional collaborative filtering approaches such as matrix factorization can be expressed as differentiable architectures ([23], [22]) and trained efficiently with Tensorflow or PyTorch, enabling efficient GPU-enabled training and free automatic differentiation.

3 LITERATURE REVIEW

In this section, I introduce the state-of-the-art research on deep learning based recommendation models.

3.1 Multilayer Perceptron-based Models

Many existing recommendation models are linear methods; thus Multilayer Perceptron (MLP) can add non-linear transformation to such models and interpret them into neural extensions. Neural Collaborative Filtering [23] and Neural Network Matrix Factorization [15] are studies that use dual neural network to model the two-way interaction between user preferences and items features. Deep Factorization Machine [20] is an end-to-end model which seamlessly integrates factorization machine and MLP. It can model

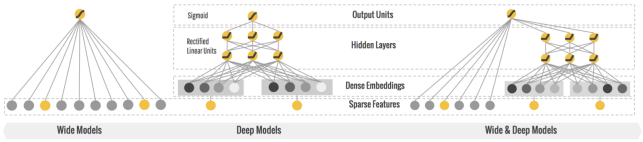

Figure 3: Wide and Deep Learning Model from Google

the high-order feature interactions via deep neural network and low-order interactions with factorization machine. In particular, the factorization machine captures the linear and pairwise interactions between features, while the MLP captures the nonlinear and deep structure between features.

MLP can also be used to represent item features directly. Wide and Deep Learning [12] is a model introduced initially by Google to solve the app recommendation problem in Google Play (Figure 3). The wide learning component is a single layer perceptron, while the deep learning component is a multilayer perceptron. The wide learning component can catch the direct features from historical data to capture memorization, while the deep learning component can produce more general and abstract representations to capture generalization. This unification can improve the accuracy and diversity of recommendation. Another work from Google applies MLP in YouTube recommendation [14], in which the recommendation task is divided into two stages: generating candidate and ranking candidate. The candidate generation network receives a subset from video corpus, then the ranking network builds a top-k list of videos based on the nearest neighbor scores from the candidates.

3.2 Autoencoder-based Models

There are currently two ways to use autoencoder to recommendation system: either (1) using it to learn lower-dimensional feature representations at the bottleneck layer or (2) filling the missing values of the interaction matrix directly in the reconstruction layer.

In the first case, Collaborative Deep Learning [52] is a hierarchical Bayesian model which integrates stacked denoising autoencoder into probabilistic matrix factorization. The Bayesian model has two components: a perception component and a task-specific component. This combination enables the model to balance the influences of side information and interaction history. Collaborative Deep Ranking [56] is a model designed specifically in a pairwise framework for top-k recommendation. Experimental results show that Collaborative Deep Ranking outperforms Collaborative Deep Learning in terms of ranking prediction. AutoSVD++ [58] makes use of contractive autoencoder [43] to learn item feature representations, then integrates them into SVD++ [26], a classic recommendation model. Compared to other autoencoders variants, contractive autoencoder captures the infinitesimal input variations; thus the model can learn the implicit feedback to further enhance the accuracy.

In the second case, Autoencoder-based Collaborative Filtering (ACF) [37] is the first auto-encoder based collaborative recommendation model. The model decomposes the partial observed vectors in the interaction matrix by integer ratings. The cost function reduces the mean squared error. However, ACF fails to deal with non-integer ratings and its decomposition of partial observed vectors

increases the sparsity of input data. Collaborative Denoising Autoencoder (CDAE) [55], a model designed for the top-k recommendation problem, is principally used for ranking prediction. CDAE learns distributed representations of the users and items via formulating the user-item feedback data using a denoising autoencoder structure. The authors proposed a negative sampling technique to sample a small subset from the negative set, which reduces the time complexity substantially without degrading the ranking quality. Multi-VAE and Multi-DAE [28] proposed a variant of Variational Autoencoder for recommendation with implicit data, showing better performance than CDAE. The authors introduced a principled Bayesian inference approach for parameters estimation and show favorable results than commonly used likelihood functions.

3.3 Restricted Boltzmann Machine-based Models

Restricted Boltzman Machine for Collaborative Filtering [45] is the principal work on this category. The visible unit of an RBM is limited to binary values, and the rating score is thus represented in a one-hot vector to adapt to this restriction. RBM is not tractable, but the parameters can be learned via the Contrastive Divergence algorithm. The authors also proposed using a conditional RBM to incorporate the implicit feedback.

The above RBM-CF is user-based since a given user's rating is clamped on the visible layer. Similarly, we can design an item-based RBM-CF by clamping the item's rating on the visible layer. [17] combines the user-based and item-based RBM-CF in a unified framework, in which the visible units are determined both by user and item hidden units. [29] designed a hybrid RBM-CF based on conditional RBM that incorporates item features.

3.4 Deep Hybrid Models

As deep neural networks are very flexible, we can easily integrate neural building blocks to build complex and powerful ensemble models. Collaborative Knowledge Based Embedding (CKE) [57] combines CNNs with Autoencoder to extract features from images. The author leverages structural, visual, and textual content using different embedding techniques. [53] exploited integrating RNNs and Denoising Autoencoder to improve model robustness when dealing with sequential data. The authors designed a generalization of RNNs called robust recurrent network. Then, they used a hierarchical Bayesian recommendation model called CRAE that can capture the sequential information of item content. CRAE has both an encoding and a decoding part, but it replaces the feed-forward neural layers with RNNs.

4 MOVIELENS1M DATASET

For my experiments, I worked with the MovieLens1M Dataset, a famous dataset within the recommendation systems research community (Figure 4). The data contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 MovieLens users who joined MovieLens in 2000. There are 3 files in the data: ratings, users, and movies.

- **Ratings**: There are 4 columns in this file - UserID, MovieID, Rating, and Timestemp. UserIDs range between 1 and 6040. MovieIDs range between 1 and 3952. Ratings are made on a 5-star scale (whole-star ratings only). Timestamp is represented in seconds. Each user has at least 20 ratings.
- **Users**: There are 5 columns in this file - UserID, Gender, Age, Occupation and Zipcodes. All demographic information is provided voluntarily by the users and is not checked for accuracy. Only users who have provided some demographic information are included in this data set. Gender is denoted by a "M" for male and "F" for female. Occupation is chosen from 21 different choices: "other", "academic / educator", "artist", "clerical / admin", "college/grad student", "customer service", "doctor/health care", "executive / managerial", "farmer", "homemaker", "K-12 student", "lawyer", "programmer", "retired", "sales / marketing", "scientist", "self-employed", "technician / engineer", "tradesman / craftsman", "unemployed", and "writer." Age is chosen from 7 different ranges: "Under 18", "18-24", "25-34", "35-44", "45-49", "50-55", and "56+".
- **Movies**: There are 3 columns in this file - MovieID, Title, and Genres. Titles are identical to titles provided by the IMDB (including year of release). Genres are pipe-separated and are selected from 18 different genres: "Action", "Adventure", "Animation", "Children's", "Comedy", "Crime", "Documentary", "Drama", "Fantasy", "Film-Noir", "Horror", "Musical", "Mystery", "Romance", "Sci-Fi", "Thriller", "War", and "Western."

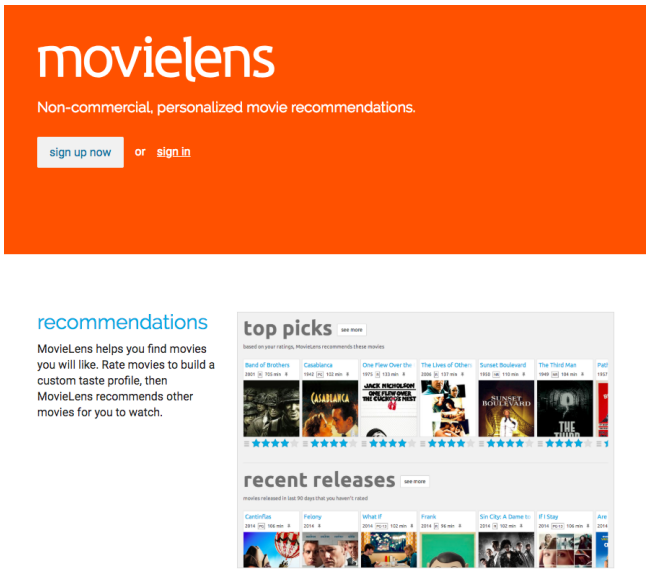

Figure 4: MovieLens 1M Dataset

5 EXPERIMENT 1 - MATRIX FACTORIZATION FOR COLLABORATIVE FILTERING

5.1 Standard Matrix Factorization

Matrix factorization in recommendation systems can be posed as:

$$R_{ui} = p_u \cdot q_i$$

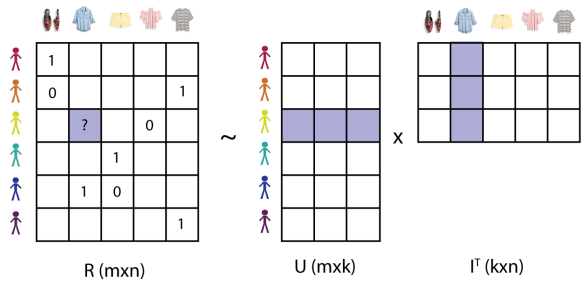

Figure 5: Understanding Latent Style from Stitch Fix

Here I am predicting the response of user u to item i with a single term: the dot product between a k-dimensional embedding vectors of the user p_u and the item q_i (*how well matched are this user and item?*).

This framing is equivalent to a decomposition of an $m \times n$, bias-corrected response matrix R into two far lower-dimension matrices $U(m \times k)$ and $I^T(k \times n)$. Unlike many machine learning algorithms, I am learning the implicit latent factors that compose the embeddings from raw responses instead of declaring them up front as features. Because I have not already observed every user-item interaction, R is sparse, as seen in Figure 5.

This formalism assumes there are only k latent factors that contribute to the preference of an user for an item, where k is chosen to best recover the information in our observed R. It is this assumption that reduces the complexity of this problem from on the order of $(m \times n)$ to the much more tractable $(k \times (m + n))$ (though at the expense of convexity).

To solve the equation above, one common choice is to add an L2 regularization penalty to each element and optimize all parameters simultaneously with stochastic gradient descent.

The baseline model above can only capture interactions. What if a user generally likes everything? What if an item is generally popular? For my next model, I added on some biases:

$$R_{ui} = b + \omega_u + \omega_i + p_u \cdot q_i$$

Here I added the global bias b, the bias of the user ω_u (*how much do they like things generally?*), and the bias of the item ω_i (*how popular is this item?*).

5.2 Advanced Matrix Factorization

What if I know more "side" features, like the occupation of the user? This is great when I want to "cold-start" a new user. I have 2 choices for side features: adding them as a bias (*artists like movies more than other occupations*) and/or adding them as a vector (*realtors love real estate shows*).

$$R_{ui} = b + d_o + \omega_u + \omega_i + (p_u + t_o) \cdot q_i$$

For this next model, I added the bias for occupation d_o (if occupation changes like rate) and the vector for occupation t_o (if occupation changes depending on the item).

What about when features change in time? For my next model, I added the user-time interactions:

$$R_{ui} = b + \omega_u + \omega_i + p_u \cdot q_i + m_u \cdot n_t$$

Here I added a set of time series n_t and m_u to pick out a few of these time series. I also would like to update the regularization penalty with total variation.

5.3 State-Of-The-Industry Matrix Factorization

Lastly, I also tried out 3 approaches that are at the frontier of the industry:

- **Factorization Machines** - which is useful when there are many kinds of interactions, not just user-item and user-time.
- **Mixture of Tastes** - which is like "attention" for recommendations and allows for users to have multiple "tastes."
- **Variational Matrix Factorization** - which optimizes a posterior instead of a point estimate, which loosely speaking expresses a spectrum of model configurations that are consistent with the data. Going variational allows me to measure what my model does not know and helps explain the data.

5.4 Evaluation

For a quick summary, here are the 7 variations of Matrix Factorization that I implemented in PyTorch for this experiment:

- **Model 1**: This is a vanilla Matrix Factorization model that serves as the baseline.
- **Model 2**: This is a Matrix Factorization model that includes biases for extra predictive power.
- **Model 3**: This is a Matrix Factorization model that adds in "side" features, which is especially useful in cold-start situations.
- **Model 4**: This is a Matrix Factorization model that includes temporal effects which can track seasonal and periodic changes.
- **Model 5**: This is a Factorization Machine model that enables a huge number of interactions while keeping computation under control.
- **Model 6**: This is a Matrix Factorization model that includes a mixture of tastes.
- **Model 7**: This is a Variational Matrix Factorization model which takes advantage of Bayesian Deep Learning and is relevant for Explore and Exploit problems.

For all 7 models, I worked with the MovieLens1M dataset, more specifically the ratings file. The ratings are split into 75% training and 25% test. All models are trained in 50 epochs with batch size of 1024 and learning rate of 0.01. The metrics to be captured are the mean squared error for the training set and the accuracy for the test set.

Table 1 captures the results. Here are some observations:

- The Matrix Factorization model that includes Mixture of Tastes (Model 6) has the lowest mean squared error for the training set. Surprisingly, the Variational Matrix Factorization model (Model 7) has the highest mean squared error.
- On the other hand, the Variational Matrix Factorization model (Model 7) also has the highest accuracy for the test

Models	Train MSE	Test Accuracy
Model1	0.748	0.835
Model2	0.670	0.787
Model3	0.619	0.788
Model4	0.760	0.793
Model5	0.658	0.821
Model6	0.609	0.817
Model7	1.762	0.836

Table 1: Mean Squared Error and Accuracy Metrics for 7 Different Matrix Factorization Models

set. The Matrix Factorization model which includes biases has the lowest accuracy.

- Most models converge after 10-15 epochs, which is the early stopping point where further training lead to minimal increase in performance.
- The more advanced the models, the longer it takes to train them. The VMF model takes the longest time to run.

6 EXPERIMENT 2 - DEEP LEARNING FOR COLLABORATIVE FILTERING

For my next experiment, I explored current state-of-the-art deep neural networks approaches for collaborative filtering.

6.1 Neural Collaborative Filtering

This model is developed in [23]. There are 3 models implemented from the paper, including Generalized Matrix Factorization (GMF), Multi-Layer Perceptron (MLP), and Neural Matrix Factorization (NeuMF).

NeuMF is essentially a fusion of GMF and MLP (Figure 6). The model takes two integers (two indices) as inputs representing the item i and user u, and output a number between 0 and 1. The output represents the probability that the user u will be interested in item i. The architecture of the Neural Network can be split into two parts: the Matrix Factorization part and the Fully-Connected part. These parts are concatenated, and then passed on to a Sigmoid Layer.

All 3 models are implemented in PyTorch and trained in 200 epochs, Adam optimizer, batch size of 1024, learning rate of 0.001, latent dimension of 8, and L2 regularization of 0.01.

6.2 Variational Autoencoders Collaborative Filtering

This model is developed in [28]. Autoencoders were initially used to learn a representation of the data (encoding). They decompose into two parts: the encoder, which reduces the shape of the data with a bottleneck, and the decoder, that transforms the encoding back into its original form. As there is a dimension reduction, the neural network will need to learn a representation in lower dimension of the input (the latent space) to be able the reconstruct the input. In the context of recommendation system, they can be used to predict new recommendation. To do so, the input and the output are both the response vector (it is usual for Autoencoders that the input and the output are the same). This means that the model will have to

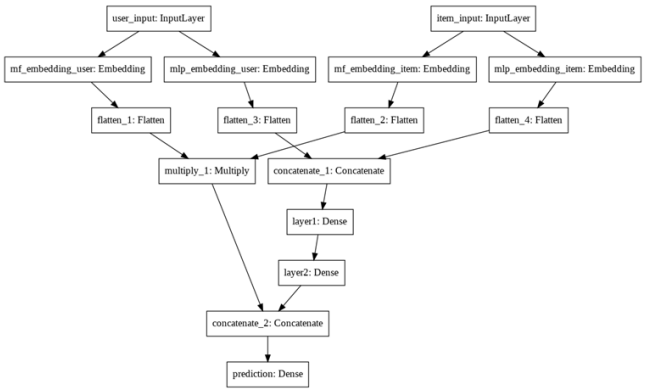

Figure 6: Neural Matrix Factorization Model

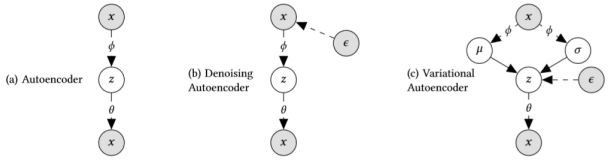

Figure 2: A taxonomy of autoencoders. The dotted arrows denote a sampling operation.

Figure 7: VAEs for Collaborative Filtering

reconstruct the response vector as some element from the input will be missing, hence learning to predict the recommendation for a given response vector.

Variational Autoencoders are an extension of Autoencoders. Instead of having a simple dense layer for the bottleneck, it will have a sampling layer. This layer will use the mean and variance from the last layer of the encoder to get a Gaussian sample and use it as input for the decoder (Figure 7).

This model is implemented in PyTorch and trained in 200 epochs, Adam optimizer, batch size of 500, learning rate of 0.0001, annealing steps of 200000 and annealing parameter of 0.2.

6.3 Restricted Boltzmann Machines Collaborative Filtering

This model is developed in [45]. Restricted Boltzmann Machines (RBM) is a stochastic neural network consisting of: (1) One layer of visible units (users' movie preferences whose states I know and set); (2) One layer of hidden units (the latent factors I try to learn); and (3) A bias unit (whose state is always on, and is a way of adjusting for the different inherent popularity of each movie) (Figure 8).

Furthermore, each visible unit is connected to all the hidden units (this connection is undirected, so each hidden unit is also connected to all the visible units), and the bias unit is connected to all the visible units and all the hidden units. To make learning easier, I restrict the network so that no visible unit is connected to any other visible unit and no hidden unit is connected to any other hidden unit.

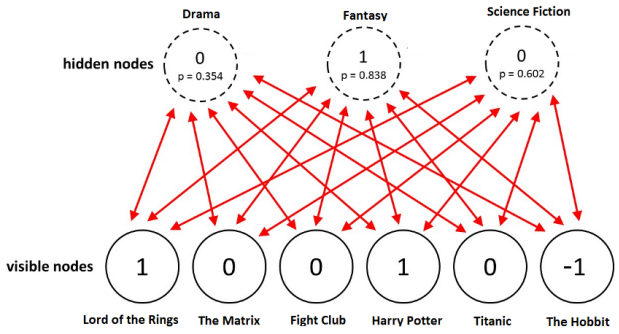

Figure 8: RBMs for Collaborative Filtering

This model is implemented in PyTorch and trained in 200 epochs, batch size of 100 and 100 hidden units.

6.4 Evaluation

For all of these deep learning models, I worked with the MovieLens1M dataset, more specifically the ratings file. The ratings are split into 75% training and 25% test. The metrics to be captured include the Normalized Discounted Cumulative Gain (NDCG) and Hit Ratio (HR). With out special mention, I truncated the ranked list at 10 for both metrics. As such, the HR intuitively measures whether the test item is present on the top-10 list, and the NDCG accounts for the position of the hit by assigning higher scores to hits at top ranks. I calculated both metrics for each test user and reported the average score.

Table 2 captures the results. The best model when looking at the NDCG is the VAE. For the Hit Ratio index, the best model is the RBM. Here are pros and cons of each model based on my experiment:

- One of the main advantages of the NMF model compared to simple Matrix Factorization is that it is non-linear, so it can capture more complex patterns in the data. However, it is easily overfitting for big datasets. Furthermore, one of the problems with this method is that, for a given user, I need to parse all the items. This can become a scalability problem when the number of items increases.
- VAE is a non-linear model that can capture complex patterns in the data. The query time is also fast because one forward pass is sufficient to get the recommendation for a given user. However, VAE is quite complex to implement, considering that the sampling layer makes it difficult to compute a gradient descent with backpropagation. Also, it is not feasible to explain the results.
- Since RBM is also a neural network, it is also non-linear and thus can capture more complex patterns in the data. Additionally, the RBM learns complex features from the data that are represented by the hidden layer. By doing some analysis in terms of genres and actors, I can technically manage to explain the results, which is a huge boost in terms of interpretability. The main disadvantage of RBM is that the training revolves around a method called Gibbs Sampling, which implies a lot of sampling and is computationally intensive.

Models	HR	NDCG
NMF	0.730	0.173
VAE	0.837	0.403
RBM	0.959	0.155

Table 2: Hit Ratio and Normalized Discounted Cumulative Gain for 3 Different Deep Learning Models

7 CONCLUSION AND FUTURE WORK

The number of research publications on deep learning-based recommendation systems has increased exponentially in the past recent years. In particular, the leading international conference on recommendation systems, RecSys, started to organize regular workshops on deep learning since 2016. For example, in the 2019 conference in Copenhagen a couple of months ago, there is a whole category of papers on deep learning, which promotes research and encourages applications of such methods.

It is easy to see why. Looking at the results from my set of experiments above, deep learning methods such as VAE or RBM can outperform classical methods like Matrix Factorization. This is because these non-linear probabilistic models enable me to go beyond the limited modeling capacity of linear factor models.

However, there are still some drawbacks of using deep neural networks for a recommendation model.

- A common objection of deep learning is that the hidden weights and activations are **hard to interpret**. Deep learning is well-known to behave like black boxes, and providing explainable predictions seem to be a really challenging task.
- Deep learning also requires **a lot of data** to fully support its rich parameterization. As compared with other domains like vision and language, it is easy to gather a significant amount of data within the context of recommendation systems research.
- Deep learning needs **extensive hyper-parameter tuning**, which is a common problem for machine learning in general.

There are 2 research directions that I am interested in doing moving forward:

- **Multi-Task Learning** is an approach in which multiple learning tasks are solved at the same time while exploiting commonalities and differences across tasks. It has been used successfully in many computer vision and natural language processing tasks.
- The single-domain recommendation system only focuses on one domain and ignores the user interests in other domains, which greatly exacerbates the sparsity and cold start problems. A tangible solution for these problems is to apply **domain adaptation** techniques, in which a model is assisted with the knowledge learned from source domains. A very popular and well-studied topic in this scenario is **transfer learning**, which can improve learning tasks in one domain by using knowledge transferred from other domains.

Several existing works indicate the efficacy of deep learning in catching the generalizations and differences across different domains and generating better recommendations on cross-domain platforms. In my opinion, this is a promising research direction but is still largely under-explored for recommendation system research in general.

At the moment, I am working with professor Ororbia to develop a Deep Boltzmann Machine model for the collaborative filtering task. The goal is to enable both multi-task learning and transfer learning. I am also planning to incorporate more datasets besides MovieLens, ideally in other domains such as music and e-commerce.

REFERENCES

[1] Asim Ansari, Skander Essegaier, and Rajeev Kohli. 2000. Internet Recommendations Systems. In *J. Marketing Research, pages 363-375.*

[2] Esma AÃŕmeur, Gilles Brassard, JosÃĺ Fernandez, and Flavien Serge Mani Onana. 2008. Alambic: a privacy-preserving recommender system for electronic commerce. In *International Journal of Information Security, volume 7, issue 5, pages 307-334.*

[3] Marko Balabanovic and Yoav Shoham. 1994. Fab: Content-Based, Collaborative Recommendation. In *ACM Comm., volume 40, no. 3, pages 66-72.*

[4] Chumki Basu, Haym Hirsh, and William Cohen. 1999. Recommendation as Classification: Using Social and Content-Based Information in Recommendation. In *Recommender Systems. Papers from 1998 Workshop, Technical Report WS-98-08, AAAI Press.*

[5] David Bennock, Eric Horvitz, Steve Lawrence, and Lee Giles. 1999. Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model Based Approach. In *Proc. International Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering.*

[6] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. 2006. Cross-Technique Mediation of User Models. In *Proceedings of International Conference on Adaptive Hypermedia and AdaptiveWeb-Based Systems, pages 21-30. Dublin.*

[7] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. 2009. Cross-representation mediation of user models. In *User Modeling and User-Adapted Interaction 19(1-2), pages 35-63.*

[8] Shlomo Berkvosky, Yaniv Eytani, Tsvi Kuflik, and Francesco Ricci. 2007. Enhancing Privacy and Preserving Accuracy of a Distributed Collaborative Filtering. In *RecSys âĂŹ07: Proceedings of the 2007 ACM conference on Recommender systems, pages 9-16. ACM Press, New York, NY, USA.*

[9] John Breese, David Heckerman, and Carl Kadie. 1998. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *Proc. 14th Conf. Uncertainty in Artificial Intelligence.*

[10] Robin Burke. 2000. Knowledge-Based Recommender Systems. In *Encyclopedia of Library and Information Systems, volume 69, supplement 32, Marcel Dekker.*

[11] John Canny. 2002. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy, pages 45-57.*

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, and Mustafa Ispir. 2016. Wide and deep learning for recommender systems. In *RecSys, pages 7-10.*

[13] Mark Claypool, Anuja Gokhale, Tim Miranda, Pavel Murnikov, Dmitry Netes, and Matthew Sartin. 1999. Combining Content-Based and Collaborative Filters in an Online Newspaper. In *Proc. ACM SIGIR '99 Workshop Recommender System: Algorithms and Evaluation.*

[14] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for YouTube recommendations. In *RecSys, pages 191-198.*

[15] Gintare Karolina Dziugaite and Daniel Roy. 2015. Neural Network Matrix Factorization. In *arXiv preprint arXiv:1511.06443.*

[16] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the 5th IEEE Conference on Data Mining (ICDM), pages 625-628.*

[17] Kostadin Georgiev and Preslav Nakov. 2013. A non-iid framework for collaborative filtering with Restricted Boltzmann Machines. In *ICML, pages 1148-1156.*

[18] Nathaniel Good, Ben Schafer, Joseph Konstan, Al Borchers, Badrul Sarwar, Jon Herlocker, and John Riedl. 1999. Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Prof. Conf. Am. Assoc. Artificial Intelligence, pages 439-446.*

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. In *MIT Press.*

[20] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In *IJCAI, pages 2782-2788.*

[21] Peng Hang, Bo Xie, Fan Yang, and Ruimin Shen. 2004. A scalable P2P recommender system based on distributed collaborative filtering. In *Expert Systems with Applications, 27(2):203-210.*

[22] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics.

[23] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *In WWW, pages 173-182.*

[24] Markus Jessenitschnig and Markus Zanker. 2009. A Generic User Modeling Component for Hybrid Recommendation Strategies. In *E-Commerce Technology, IEEE International Conference, pages 337-344.*

[25] Alfred Kobsa. 2007. Generic User Modeling Systems. In *P. Brusilovsky, A. Kobsa,W. Nejdl (eds.) The Adaptive Web, Lecture Notes in Computer Science, volume 4321, pages 136-154. Springer.*

[26] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD, pages 426-434.*

[27] Shyong Lam, Dan Frankowski, and John Riedl. 2006. Do You Trust Your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems. In *G. MÂĺuller (ed.) ETRICS, Lecture Notes in Computer Science, volume 3995, pages 14-29. Springer.*

[28] Dawen Liang, Rahul Krishman, Matthew Hoffman, and Tony Jebara. 2018. Variational autoencoders for collaborative filtering. In *arXiv preprint arXiv:1802.05814.*

[29] Xiaomeng Liu, Yuanxin Ouyang, Wenge Rong, and Zhang Xiong. 2015. Item Category Aware Conditional Restricted Boltzmann Machine Based Recommendation. In *International Conference on Neural Information Processing, pages 609-616.*

[30] Qusay H. Mahmoud. 2006. Provisioning Context-Aware Advertisements to Wireless Mobile Users. In *Multimedia and Expo, IEEE International Conference on pages 669-672.*

[31] Lorraine McGinty and Barry Smyth. 2003. On the Role of Diversity in Conversational Recommender Systems. In *A. Aamodt, D. Bridge, K. Ashley (eds.) ICCBR 2003, the 5th International Conference on Case-Based Reasoning, pages 276-290. Trondheim, Norway.*

[32] David McSherry. 2002. Diversity-Conscious Retrieval. In *S. Craw, A. Preece (eds.) Advances in Case-Based Reasoning, Proceedings of the 6th European Conference on Case Based Reasoning, ECCBR 2002, pages 219-233. Springer Verlag, Aberdeen, Scotland.*

[33] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the Netflix Prize contenders. In *KDD âĂŹ09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 627-636. ACM, New York, NY, USA.*

[34] Mari Carmen Puerta Melguizo, Lou Boves, and Olga Munoz Ramos. 2007. A proactive recommendation system for writing: Helping without disrupting. In *ECCE âĂŹ07: Proceedings of the 14th European conference on Cognitive ergonomics, pages 89-95. ACM, New York, NY, USA.*

[35] Prem Melville, Raymond Mooney, and Ramadass Nagarajan. 2002. Content-Boosted Collaborative Filtering for Improved Recommendations. In *Proc. 18th Nat'l Conf. Artificial Intelligence.*

[36] Miquel Montaner, Beatriz LÃspez, and Josep LluÃŋs de la Rosa. 2003. A Taxonomy of Recommender Agents on the Internet. In *Artificial Intelligence Review 19(4), pages 285-330.*

[37] Yuanxin Ouyang, Wenqi Liu, Wenge Rong, and Zhang Xiong. 2014. Autoencoder-based collaborative filtering. In *International Conference on Neural Information Processing, pages 284-291.*

[38] Manos Papagelis, Ioannis Rousidis, Dimitris Plexousakis, and Elias Theoharopoulos. 2005. Incremental Collaborative Filtering for Highly-Scalable Recommendation Algorithms. In *M.S. Hacid, N.V. Murray, Z.W. Ras, S. Tsumoto (eds.) ISMIS, Lecture Notes in Computer Science, vol. 3488, pages 553-561. Springer.*

[39] Michael Pazzani. 1999. A Framework for Collaborative, Content-Based, and Demographic Filtering. In *Artificial Intelligence Rev., pages 393-408.*

[40] Huseyin Polat and Wenliang Du. 2003. Privacy-Preserving Collaborative Filtering Using Randomized Perturbation Techniques. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA, pages 625-628.*

[41] Alexandrin Popescul, Lyle Ungar, David Pennock, and Steve Lawrence. 2001. Probabilistic Models for Unified Collaborative and Content-Based Recommendation in Sparse-Data Environments. In *Proc. 17th Conf. Uncertainty in Artificial Intelligence.*

[42] Naren Ramakrishnan, Benjamin Keller, Batul Mirza, Ananth Grama, and George Karypis. 2001. When being Weak is Brave: Privacy in Recommender Systems. In *IEEE Internet Computing.*

[43] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. 2011. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML, pages 833-840.*

[44] Somkiat Sae-Ueng, Sineenard Pinyapong, Akihiro Ogino, and Toshikazu Kato. 2008. Personalized Shopping Assistance Service at Ubiquitous Shop Space. In *IEEE 22nd International Conference on pages 838-843.*

[45] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for collaborative filtering. In *ICML, pages 791-798.*

[46] Badrul Sarwar, Joseph Konstan, and John Riedl. 2005. Distributed Recommender Systems for Internet Commerce. In *M. Khosrow-Pour (ed.) Encyclopedia of Information Science and Technology (II), pages 907-911. Idea Group.*

[47] Andrew Schein, Alexandrin Popescu, David Pennock, and Lyle Ungar. 2002. Methods and Metrics for Cold-Start Recommendations. In *Proc. 25th Ann. Int'l ACM SIGIR Conf.*

[48] Upendra Shardanand and Pattie Maes. 1995. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In *Prof. Conf. Human Factors in Computing Systems.*

[49] Reza Shokri, Pedram Pedarsani, George Theodorakopoulos, and Jean-Pierre Hubaux. 2009. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys âĂŹ09: Proceedings of the third ACM conference on Recommender systems, pages 157-164. ACM, New York, NY, USA.*

[50] Ian Soboroff. 1999. Combining Content and Collaboration in Text Filtering. In *Proc. Int'l Joint Conf. Artificial Intelligence Workshop: Machine Learning for Information Filtering.*

[51] Nima Taghipour and Ahmad Kardan. 2008. A hybrid web recommender system based on Q-learning. In *Proceedings of the 2008 ACM symposium on applied computing, pages 1164-1168. ACM, New York, NY, USA.*

[52] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD, pages 1235-1244.*

[53] Hao Wang, Shi Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS, pages 415-423.*

[54] Yang Wang and Alfred Kobsa. 2009. Performance Evaluation of a Privacy-Enhancing Framework for Personalized Websites. In *G.J. Houben, G.I. McCalla, F. Pianesi, M. Zancanaro (eds.) UMAP, Lecture Notes in Computer Science, volume 5535, pages 78-89. Springer.*

[55] Yao Wu, Christopher DuBois, Alice Zheng, and Martin Ester. 2016. Collaborative denoising autoencoders for top-n recommender systems. In *WSDM, pages 153-162.*

[56] Haochao Ying, Liang Chen, Yuwen Xiong, and Jian Wu. 2016. Collaborative deep ranking: a hybrid pair-wise recommendation algorithm with implicit feedback. In *PAKDD, pages 555-567.*

[57] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *SIGKDD, pages 353-362.*

[58] Shuai Zhang, Lina Yao, and Xiwei Xu. 2017. AutoSVD++: An Efficient Hybrid Collaborative Filtering Model via Contractive Autoencoders.

[59] Yongfeng Zhang, Qingyao Ai, Xu Chen, and Bruce Croft. 2017. Joint representation learning for top-n recommendation with heterogenoues information sources. In *In CIKM, pages 1449-1458.*