UNIVERSITY OF ILLINOIS AT CHICAGO

A Survey and Critique of Deep Learning on Recommender Systems

by

Lei Zheng

in the

DEPARTMENT OF COMPUTER SCIENCE

September 2016

Abstract

Recommender systems have become extremely common in recent years. Companies, such as *Amazon* or *eBay*, developed a large number products to meet different needs of customers. A increasing number of options are available to customers in the era of E-commerce. Thus, in this new level of customization, in order to find what they really need, customers must process a large amount of information provided by businesses. One solution to ease this overload problem is recommender systems. On one hand, traditional recommender systems recommend items based on different criteria, such as the past preference of users or user profiles. On the another hand, deep learning techniques achieve promising performance in various areas, such as Computer Vision, Audio Recognition and Natural Language Processing. However, applications of deep learning in recommender systems have not been well explored yet. In this article, we firstly introduce traditional techniques involved in recommender systems and deep learning in the first chapter. And then, a survey and critique of several state-of-the-art deep recommendation systems will be provided in the following chapters.

Contents

Chapter 1

Introduction

1.1 Recommender Systems

During the last decade, the variety and number of products and services provided by companies has increased dramatically. Companies produce a large number of products to meet the needs of customers. Although this gives more options to customers, it makes it harder for them to process the large amount of information provided by companies. Recommender systems are designed to help customers by introducing products or services. These products and services are likely prefered by them based on user preferences, needs, and purchase history. Nowadays, many people use recommender systems in their daily life such as online shopping, reading articles, and watching movies.

Given a set of users U and a set of items V, a recommender systems is designed to recommend items to the users according to the their purchase history or past ratings. Usually, a recommeder system recommends items by either predicting ratings or providing a ranked list of items for each user. And, two kinds of techniques involved in recommender systems are: Collaborative Filtering and Content-based recommendations.

1.1.1 Collaborative Filtering

Collaborative Filtering (CF) technique is a popular and well-known technique involved in recommender systems. Many of the prominent recommendation approaches, such as [1–3], are based on Collaborative Filtering (CF) technique [4].

Collaborative Filtering (CF) follows a simple observation that users tend to buy items preferred by users with similar tastes. For example, in Table 1.1, user U_1 tends to buy item I_2 since both user U_1 and U_4 prefer I_1 and user U_4 gives high rating for item I_2.

In a common Collaborative Filtering (CF) setting, there is a collection of preferences from users. For example, a list of M users $\{u_1, u_2, ..., u_M\}$ and a list of N items $\{i_1, i_2, ..., i_N\}$ are given. A list of items, i_{u_i}, has been rated by user u_i. The ratings can either be explicit indications on a 1-5 scale, or implicit indications. Implicit indications are generally implicit feedback, such as purchases or click-throughs, from users. CF techniques can be either memory-based or model-based.

1.1.1.1 Memory-based CF

In memory-based CF systems, recommendations or predictions are made based on similarity values. Rating data is used to calculated the similarity or weight between users or items.

There are several advantages of memory-based CF. First of all, since we only need to calculate similarity, it is easy to implement. Second, memory-based CF systems are scalable to large size data. Third, most of memory-based systems are online learning models. Thus, new arrival data can be handled easily. At last, the recommendation results can be understood and can provide feedback to explain why recommend these items. However, several limitations are also existed in memory-based CF techniques. For example, since the similarity values are based on common items, when data are sparse and common rated items are very few, the recommendation results are unreliable and not accurate.

Neighbor-based CF is one of the most representative memory-based CF models. Neighbor-based CF involves in two steps: similarity calculation and prediction. In the similarity calculation step, the similarity values can be measured between users or items.

For example, the *pearson correlation* [5] between two users u and v is as following:

$$w_{u,v} = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}} \tag{1.1}$$

where the $i \in I$ sums over the items that are rated by both user u and user v.

Vector Cosine-based Similarity is another similarity metric used to measure the difference between documents. Documents are represented as a vector of word frequency. In neighbor-based CF, *Vector Cosine-based Similarity* is adopted to compute the similarity across users or items. As shown in Eq. 1.2, *Vector Cosine-based Similarity* between

TABLE 1.1: An example of rating matrix

	I_1	I_2	I_3	I_4
U_1	4	?	5	5
U_2	4	2	1	
U_3	3		2	4
U_4	4	4		
U_5	2	1	3	5

item i and item j can be derived.

$$w_{i,j} = cos(\overrightarrow{i}, \overrightarrow{j}) = \frac{\overrightarrow{i} \bullet \overrightarrow{j}}{\left\|\overrightarrow{i}\right\| * \left\|\overrightarrow{j}\right\|} \tag{1.2}$$

where $\bullet$ denotes the dot product of two vectors.

In neighbor-based CF, to generate recommendations or predictions for user u, we use similarity to generate a set of users that are close to user u. Then, the prediction of user u can be calculated by using ratings from the set of users [6].

1.1.1.2 Model-based CF

Model-based CF, which is based on machine learning or data mining models, finds complex rating patterns in training data. After the training process, model-based CF models are expected to make intelligent predictions or recommendations for users. Model-based CF algorithms are developed to counter the shortcomings of memory-based CF models.

One of simple model-based CF models to make recommendations is Naive Bayes (NB). In NB, we assume that features are independent with each other. When we apply NB to recommder systems, a similar assumption is used. For example, as shown in Table 1.1, to derive the probability of what rating user U_1 will give to item I_2, we can calculate as Eq. 1.3

$$\begin{aligned} rating &= argmax \quad p(r|U_2 = 2, U_4 = 4, U_5 = 1) \\ &= argmax \quad p(r)p(U_2 = 2|r)p(U_4 = 4|r)p(U_5 = 1|r) \end{aligned} \tag{1.3}$$

Several researchers conduct experiments to exploit potential of NB in recommender systems. In the work of [7], NB is used to solve the recommendation problem in a binary rating matrix. They show that better recommendations than neighbor-based CF systems can be achieved by a simple NB based recommender systems. In addition,

[8] validates that Bayesian Network Classifier is also effective to the recommendation problem.

Clustering algorithms are also useful in recommender systems. Clustering algorithms are unsupervised learning algorithms and designed to cluster objects into different categories without label information. In CF, clustering usually can be used as an intermediate step. First, one clustering algorithm, such as K-Means, is used to cluster users or items into different groups. Then, the conditional probability of ratings for an item can be calculated based on their group information.

The most successful model-based CF techniques is Matrix Factorization (MF) [1]. They find common factors that can be the underlying reasons of the ratings given by users. For example, in a movie recommender system, these factors can be genre, actors, or director of movies that may affect the rating behavior of users. Matrix factorization techniques not only find these hidden factors, but also give the importance of them for each user and how each item satisfies each factor. Matrix factorization techniques get the matrix containing all the available ratings and find a feature set for each user and item as the result of the factorization process. Then, a rating that each user assigns to each item can get estimated by the scalar product of the two feature vectors corresponding to those user and item. In this way, users with similar preference will have similar latent features, and items which are favored by similar users will share similar latent features.

As shown in Eq. 1.4, MF approximates the rating matrix R with two matrices: U and V, which can be viewed as latent factors of users and items, respectively. Each row of matrix U and each column of matrix V are the latent factors of a user or item respectively. By multiplying latent factors of a user to latent factors of an item, we get an estimation of the corresponding rating.

$$R \approx U \times V \tag{1.4}$$

[9] presents Probabilistic Matrix Factorization (PMF) which extends MF into the probabilistic framework. Instead of approximating the rating matrix R by using two low-rank matrices U, V, PMF models matrices U, V with two Gaussian distributions. As seen in Fig. 1.6, user and item matrices U and V are drew from Gaussian distributions. The probability of rating matrix R can be calculated as Eq/ 1.5

$$p(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[N(R_{ij}|U_i^T V_j, \sigma^2) \right]^{I_{ij}} \tag{1.5}$$

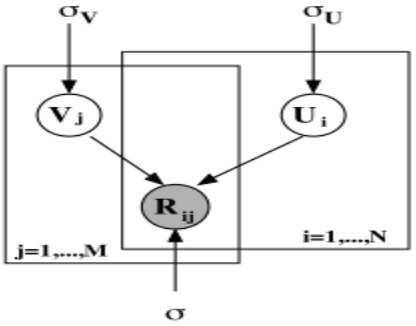

FIGURE 1.1: The Graphic Model for PMF

Thus, the learning process of PMF is that given ratings R and hyper-parameters $\{\sigma^2, \sigma_V^2, \sigma_U^2\}$, maximize the log-posterior as Eq. 1.6

$$\begin{aligned} lnp(U,V|R,\sigma^2,\sigma_U^2,\sigma_V^2) = \frac{1}{2\sigma^2}\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}(R_{ij}-U_i^T V_j)^2 - \frac{1}{2\sigma_U^2}\sum_{i=1}^{N} U_i^T U_i - \frac{1}{2\sigma_V^2}\sum_{j=1}^{M} V_j^T V_j \\ -\frac{1}{2}\left(\left(\sum_{i=1}^{N}\sum_{j=1}^{M} I_{ij}\right) ln\sigma^2 + NDln\sigma_U^2 + MDln\sigma_V^2\right) + C \end{aligned} \tag{1.6}$$

Although PMF achieves better performance than MF, similar to MF, PMF only uses rating data and fails to utilize content information to deal with the sparsity problem. The lack of ratings hinder PMF to produce high quality recommendations for users with few ratings. In addition, PMF is still a shallow model and unable to capture complex rating patterns existing in rating data.

1.1.2 Content-based Recommender Systems

Although model-based CF achieves great success and attracts a lot attention, model-based CF still has some drawbacks. The most important one is that most of model-based CF models suffer from the sparsity problem. For those users who provide no ratings, model-based CF is unable to generate reasonable recommendations.

To deal with the sparsity problem, researchers propose Content-based Recommender Systems. These approaches utilize different resources, such as item information or user profiles, to learn latent factors of users or items. In this manner, even one user provides little ratings, the preference of the user can still be inferred.

Many existing works [10–12] only separately studied ratings and reviews. Studies on reviews often concentrate on analyzing product features and sentiment [13, 14]. One of

the pioneer works that explored using reviews to improve the rating prediction is done by [15]. It found that reviews are usually related to different aspects, such as price, service, positive or negative feelings, and these aspects can be exploited for rating prediction.

Further works [16, 17] attempt to discover aspects from review text by using topic modeling in an unsupervised manner. In [18], the authors propose to employ Topic Modeling to discover latent aspects from either item or user reviews and predict ratings in an unified model. They adopt an objective function that combines the accuracy of rating prediction and the likelihood of the review corpus. This method achieves significant improvement, compared with models which only use ratings or reviews. Similar to [18], [19] also adopts Topic Modeling to discover latent factors from item reviews but employs a mixture of Gaussian to model ratings.

Collaborative Topic Modeling [20] is designed to recommend articles. User and item latent factors are modeled by a Gaussian distribution while item reviews are generated from a topic distribution. Since both user reviews and item reviews reflect the characteristics and latent aspects of corresponding users and items, [21] leverages user information with a combination between collaborative filtering and aspect based opinion mining. However, unlike we learn features automatically from reviews, it use hand-crafted textual or user-relevant features. [22] proposes a probabilistic model based on collaborative filtering and topic modeling. This approach uncovers aspects and sentiments of users and items. But, it does not incorporate ratings during modeling reviews.

[23] simultaneously exploits ratings and user and item reviews. But, this method suffers from a scalability problem and can not deal with new coming users and items. Overall, a limitation of the above studies is that their textual similarity is solely based on lexical similarity. If two reviews are semantically related but use different words, these models may not consider the two reviews to be similar. As we know, the vocabulary in English is very diverse and two reviews can be semantically similar even with low lexical overlap, so semantic meaning is especially important and has been lost in the works above. What is more, in these works, reviews are represented by using *bag-of-words*. Therefore, word order existed in reviews have not been preserved.

1.2 Deep Learning

Most of machine learning algorithms, such as Support Vector Machine (SVM) or Logistic Regression, are of shallow architectures. Typically, shallow models consists of one or two layers. Although these models achieve good performance and dominant in the 90's,

due to its limited representation learning capacity, they have difficulties in modeling complicated data, such as text, images and audios.

Recently, experimental results suggest that in order to train better AI models, deep architecture is needed. Before that, models with two or three layers at most perform better than deep models. Deep models tend to give worse results and become harder to train. Until 2006, [24] shows that with a layer-wise training strategy, a Deep Belief Network (DBN) can be successfully trained to predict hand written digits. This is the first attempt and success to train a deep model. Before that, researchers have not seriously exploited deep models due to lack of data and computational power. Generally, deep architecture models consist of multiple layers and can learn a hierarchy of features from low-level features to high-level ones.

A DBN is formed with a stack of Restricted Boltzmann Machines (RBM). In its first two layer, we train a two layer RBM with one visible layer and one hidden layer. Then, the activation probabilities of the hidden layer forms a visible layer to learn another hidden layer. In this manner, RBM can be stacked to learn a multi-layer DBN.

Another type of deep models are Deep Neural Networks (DNN). DNN is Multi-Layer Perceptron (MLP) with many hidden layers. Back-propagation (BP) [25] is employed to learn DNN. The success of DNN is due to two techniques: a larger number of hidden units and better parameter initialization techniques. A DNN with large number of hidden units can have better modeling power. Even the learned parameters of the DNN is a local optimal, the DNN can performs much better than those with less hidden units. However, in order to converge to a local optima, a DNN with large number of units also requires more training data and more computational power. This also explains why DNN becomes popular until recently. Leaning a DNN is a highly non-convex problem, It is no doubt that better parameter initialization can lead to better performance. Researchers [26] found that parameters of DNN can be initialized with the learned parameters of a DBN with the same architecture.

Deep Auto Encoder (DAE) is a special type of DNN. The difference between DAE and DNN is that DAE is an unsupervised learning algorithm and the input and output of DAE are the same. By forcing the input and output to be the same, the output of the middle layer can be regarded as dense representations. Similar to DNN, DAE can also be pre-trained by using DBN. [27] proposed a pre-training technique to learn a "deep" autoencoders with multiple layers. This technique involves treating each neighboring set of two layers as a restricted Boltzmann machine. In this manner, the pre-training procedure approximates a good parameter initialization. Then, they use a back-propagation technique to fine-tune the pre-trained model.

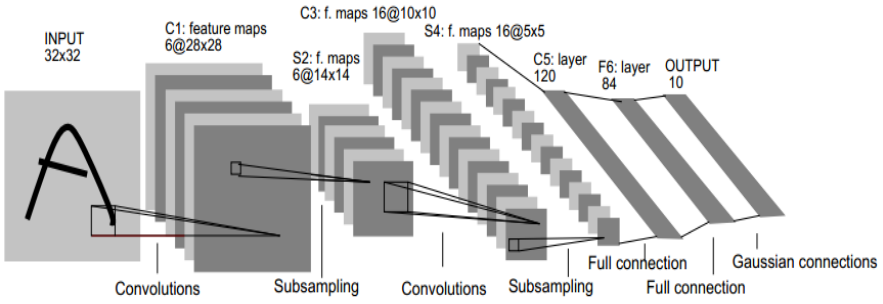

FIGURE 1.2: The architecture of CNN

Convolutional Neural Network (CNN) is one of the most successful deep models in the application of Computer Vision and are biologically-inspired variants of MLPs. As shown in Fig. 1.2, the main components of CNN are convolutional layers and subsampling layers. In its convolutional layers, the outputs of the previous layer are fed into a a set of convolutional filters and generate a set of filtered results. Then, these results are subsampleed based on their activations in a following sub-sampling layer. Convolutional layers and sub-sampling layers can be alternatively added to build a deep CNN model. As a class of deep models for learning features, the Convolutional Neural Networks (CNN) learns a hierarchy of increasingly complex features. Without building handcrafted features, these methods utilize layers with convolving filters that are applied on top of pre-trained word embeddings. Moreover, in benefiting from the shared weights, CNNs have fewer parameters than traditional feed-forward neural networks.

Deep Learning technique [24, 27–30] is a hot and emerging area in both data mining and machine learning communities. These models can be trained by either supervised or unsupervised approaches. Deep learning models are initially applied to the field of Computer Vision and Audio, Speech, and Language Processing. It outperformed many state-of-the-art models [24, 28, 31, 32]. Later deep models have shown their effectiveness for various NLP tasks. These tasks include semantic parsing [33], machine translation [34], sentence modeling [35] and a variety of traditional NLP tasks [36].

Deep learning has recently been proposed for building recommender systems for both collaborative and content based approaches. In [37], it shows that a Restricted Boltzmann Machines (RBM) model can slightly outperforms Matrix Factorization. In [38] and [39], deep models are introduced to learn embedding from music. Due to the success of Multi-view deep learning [40, 41], some researchers propose to learn latent factors from different sources. [42] builds a multi-view deep model to learn a rich feature set for users from their web browsing history and search queries. But, all approaches above ignore review text.

To utilize information from text, [43] integrates an autoencoder and PMF. Item text is represented by using *bag-of-words* and taken as input to the autoencoder to learn item features. User features are modeled by a Gaussian distribution. However, they do not jointly model users and items from text. In addition, this approach is only suitable for *one-class collaborative filtering problems* [44].

Chapter 2

Restricted Boltzmann Machines for Collaborative Filtering

In the work of [37], they introduce a two-layer Restricted Boltzmann Machines (RBM) to model ratings. Since maximum likelihood estimation is intractable in these models, they show that optimization can be done efficiently by following an approximation to the gradient of a different objective function.

2.1 The Model

Here, we suppose that we have N users, M items and ratings ranging from 1 to K. If each user rated all items, each user can be used as a training example for a RBM with M visible softmax units. And, these visible units are fully connected to a set of hidden units. However, since only a few items are rated by one user in a recommender system, most of ratings are missing. To solve this problem, the authors propose to use a different RBM for each user. An RBM only has visible softmax units for the items rated by the corresponding user, which means an RBM has few connections if the user rated only a few movies. Figure 2.1 shows an RBM for one user. As we can see, only those rated items have connections with hidden units. If user a and user b rated the same item, their RBMs share weights connected to the corresponding visible unit.

Given a set of hidden units $\mathbf{h}$, the observed binary rating matrix $\mathbf{V}$ can be derived as:

$$p(v_i^k = 1|\mathbf{h}) = \frac{exp(b_i^k + \sum_{j=1}^{F} h_j W_{ij}^k)}{\sum_{l=1}^{K} exp(b_i^l + \sum_{j=1}^{F} h_j W_{ij}^k)} \tag{2.1}$$

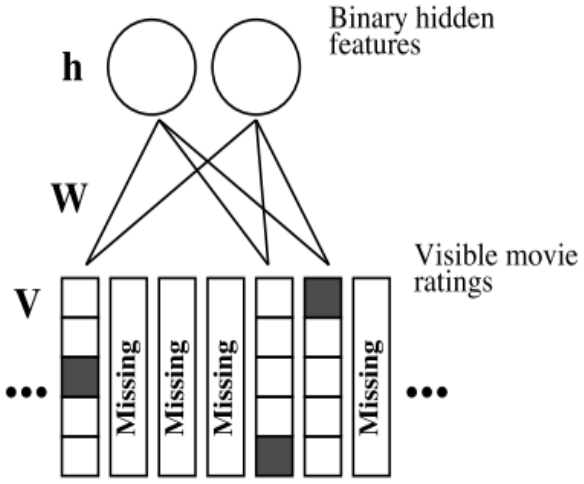

FIGURE 2.1: Architecture of one RBM

where b_i^k is is the bias term of rating k for item i, h_j is the value of j_{th} hidden unit, W_{ij}^k is the weight connecting rating k of item i and j_{th} hidden unit.

Similarly, given a set of visible units $\mathbf{v}$, the hidden units $\mathbf{h}$ can be derived as:

$$p(h_j|\mathbf{v}) = \sigma(b_j + \sum_{i=1}^{m}\sum_{k=1}^{K} v_i^k W_{ij}^k) \tag{2.2}$$

where $\sigma(x)$ is the logistic function.

2.2 Learning

In the learning procedure, gradient ascent is employed to maximize the log-likelihood. The gradients can be obtained as Eq. 2.3:

$$\begin{aligned} \Delta W_{ij}^k &= \epsilon \frac{\partial log p(V)}{\partial W_{ij}^k} \\ &= \epsilon \left(\left\langle v_i^k h_j \right\rangle_{data} - \left\langle v_i^k h_j \right\rangle_{model} \right) \end{aligned} \tag{2.3}$$

where ϵ is the learning rate. In Eq. 2.3, $\left\langle v_i^k h_j \right\rangle_{data}$ is defined as, in the data, the frequency of both item i with rating k and j_{th} hidden unit turned on together. $\left\langle v_i^k h_j \right\rangle_{model}$ is defined as the expectation with respect to the distribution of the model. However, computing $\left\langle v_i^k h_j \right\rangle_{model}$ directly is intractable. To avoid compute $\left\langle v_i^k h_j \right\rangle_{model}$, they propose to approximate the gradient of the objective function by using "Contrastive Divergence

[45]" as following:

$$\Delta W_{ij}^k = \epsilon \left(\left\langle v_i^k h_j \right\rangle_{data} - \left\langle v_i^k h_j \right\rangle_T \right) \tag{2.4}$$

where $\langle v_i^k h_j \rangle_T$ denotes samples from running Gibbs sampler by using Eq. 2.1 and Eq. 2.2 for T steps.

2.3 Prediction

Given observed ratings $\mathbf{V}$, we can employ Eq. 2.2 to calculate the states of its hidden units and then apply the following formula to calculate the expected value for a movie i:

$$p_i = \sum_k p(v_i^k = 1|h)k \tag{2.5}$$

2.4 Critique

Although the authors show that RBM can be successfully applied to the recommendation problem and slightly outperforms traditional Matrix Factorization, the proposed model is not deep enough and only consists of 2 layers. Learning deep models has been successfully applied in the domain of modeling temporal data [46] and learning word embedding [47, 48]. Training a deeper RBM is helpful for capturing hierarchical latent factors of users and items and more accurately modeling ratings.

RBM does not make use of content information, such as user profiles or review texts. RBM is typically used with rating data where most ratings are missing and take advantage of this fact for computational tractability. As a result, the proposed model can not deal with the cold start problem [49], where recommender systems are required to give recommendations to novel users who have no preference on any items, or recommending items that no user of the community has rated yet. However, content information is proved to be effective to reduce the cold start problem [49].

Chapter 3

Collaborative Deep Learning for Recommender Systems

To address the cold start problem, [43] introduces Collaborative Deep Learning (CDL) to utilize review texts and ratings. [43] introduces CDL to integrate a bayesian Stack Denoise Auto Encoder (SDAE) [50] and Collaborative Topic Regression (CTR) [20] learn latent factors of items from review texts and draw a latent user vector from Gaussian distribution.

3.1 Collaborative Deep Learning

To learn from review text of each item, CDL represents its reviews by using *bag-of-words* scheme. All J items is represented by a $J \times S$ matrix $\mathbf{X}_c$, where the j_{th} row of $\mathbf{X}_c$ is the *bag-of-words* vector $\mathbf{X}_{c,j*}$ of vocabulary size S for item j. As shown in Fig. 3.1, these vectors can be fed into a bayesian SDAE network to learn item latent factors. The input $\mathbf{X}_{0,j*}$ and output $\mathbf{X}_{c,j*}$ of the network are forced to be the same. Thus, the middle layer of the network can be viewed as a compressed representation of the items v_j. Simultaneously, latent factors of user u_i is drew from a Gaussian distribution and the rating predictions are modeled by another Gaussian distribution with the mean of ${u_i}^T v_j$. The generative process of CDL can be defined as:

1. For each layer of the SDAE network
   (a) For each column n of the weight matrix $\mathbf{W}_l$, draw $\mathbf{W}_{l,*n}$ from $N(0, \lambda_w^{-1}\mathbf{I}_{K_l})$
   (b) Draw the bias term b_l from $N(0, \lambda_w^{-1}\mathbf{I}_{K_l})$.
   (c) For each row j of $\mathbf{X}_l$, draw $\mathbf{X}_{l,j*}$ from $Delta(\sigma(X_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l))$.

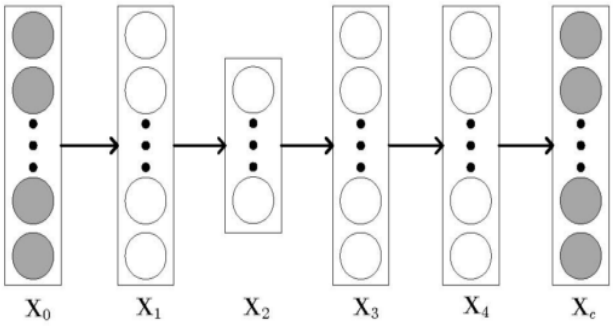

FIGURE 3.1: A 4-layer SDAE

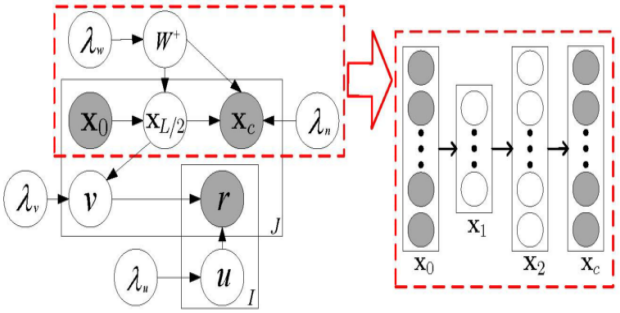

FIGURE 3.2: Architecture of CDL

2. For each item j
   (a) Draw a clean input $X_{c,j*}$ from $N(\mathbf{X}_{L,j*}, \lambda_n^{-1}\mathbf{I}_J)$.
   (b) Draw a latent item offset vector ϵ_j from $N(0, \lambda_v^{-1}\mathbf{I}_K)$ and then set the latent item vector as: $v_j = \epsilon_j + \mathbf{X}_{L/2,j*}^T$.
3. Draw a latent user vector for each user i: u_i from $N(0, \lambda_u^{-1}\mathbf{I}_K)$
4. Draw a rating r_{ij} from $N({u_i}^T v_j, c_{ij}^{-1})$.

3.2 Learning the Parameters

To train CDL, we have two sets of parameters to be optimized: parameters of the network $\{\mathbf{W}_l, b_l\}$ and latent factors $\{u_i, v_j\}$. Note that latent factor of items are the outputs of the middle layer $\mathbf{X}_{L/2}$. Thus, the middle layer $\mathbf{X}_{L/2}$ serves as a bridge to connect the network and rating modeling.

Given hyper parameters $\{\lambda_u, \lambda_v, \lambda_w, \lambda_n\}$, the log likelihood of CDL can be derived as:

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|u_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|W_l\|_F^2 + (\|b_l\|_2^2) - \frac{\lambda_v}{2}\sum_j \left\|v_j - f_e(X_{0,j*}, W^+)^T\right\|_2^2 \\ &-\frac{\lambda_n}{2}\sum_j \left\|f_r(X_{0,j*}, W^+) - X_{c,j*}\right\|_2^2 - \sum_{i,j}\frac{c_{i,j}}{2}(r_{i,j} - u_i^T v_j)^2\end{aligned} \tag{3.1}$$

where $-\frac{\lambda_u}{2}\sum_i \|u_i\|_2^2$ and $-\frac{\lambda_w}{2}\sum_l (\|W_l\|_F^2 + (\|b_l\|_2^2)$ are regularization terms, the square error between latent factors of items v_j and the output of the middle layer $f_e(X_{0,j*}, W^+)$ is minimized by $-\frac{\lambda_v}{2}\sum_j \left\|v_j - f_e(X_{0,j*}, W^+)^T\right\|_2^2$. To force the input and output of the network to be the same, $-\frac{\lambda_n}{2}\sum_j \left\|f_r(X_{0,j*}, W^+) - X_{c,j*}\right\|_2^2$ is introduced. In addition, $-\sum_{i,j}\frac{c_{i,j}}{2}(r_{i,j} - u_i^T v_j)^2$ approximates ratings by the inner product of u_i and v_j.

To maximize the log-likelihood of CDL, an EM-style algorithm has been proposed. First, we fix $\{\mathbf{W}_l, b_l\}$, u_i and v_j can be optimized by coordinate ascent. We compute the gradients of $\mathscr{L}$ with respect to u_i and v_J and set them to zero, leading to the following updating rules 3.2 and 3.3:

$$u_i \leftarrow (VC_iV^T + \lambda_u I_K)^{-1} VC_iR_i \tag{3.2}$$

$$v_j \leftarrow (UC_iU^T + \lambda_v I_K)^{-1}(UC_jR_j + \lambda_v f_e(X_{0,j*}, W^+)^T) \tag{3.3}$$

Second, fixing u_i and v_j, they use the back-propagation to learned weights and bias: $\{\mathbf{W}_l, b_l\}$.

3.3 Critique

CDL is the first deep model to learn from review texts for recommender systems. It seemly integrates an Auto Encoder and CTR to model ratings. However, CDL still has several shortcomings.

First, CDL only models item review texts. In recommender systems, user provides reviews to express their feelings. These review texts can be utilized to learn preference of users.

Second, review texts are represented by using *bag-of-words* scheme. As we know, *bag-of-words* vectors only convey the frequency of words. If two reviews are semantically related but use different words, CDL, which uses *bag-of-words*, may not consider the two reviews to be similar. The vocabulary in English is very diverse and two reviews can be semantically similar even with low lexical overlap, so semantic meaning is especially

important. However, semantic meanings, which are essential for reveal user attitudes and item properties, are lost in CDL.

At last, in CDL, word order is ignored. However, in many text modeling applications, word order is extremely important [51]. To further improve the performance of CDL, word order should be taken into consideration while modeling review texts.

Chapter 4

Deep Content-based Music Recommendation

It is well known that Collaborative Filtering can generally outperform content-based methods [52]. However, it is not valid when recommending items that have not been consumed before. To alleviate the cold start problem in music recommendation, some recommender systems recommend music based on metadata, such as genre, artist and album. But, the recommendation results are predictable and not useful. Recommender systems should recommend items that users are unknown of. A better approach is to analyze music signals to recommend similar songs users have previously listened to. To capture high level features from music signals, the authors introduce a deep CNN model to learn latent factors from music signals.

4.1 Weighted Matrix Factorization

In a music system, we only know what songs a user has listened to and how many times the user has played a song. This is implicit feedback and users never provide explicit rating. This setting is distinct with traditional matrix factorization. Traditional matrix factorization is designed to predict ratings. Weighted Matrix Factorization (WMF) [53] is a modified matrix factorization and for implicit datasets. WMF introduces two variables:

$$p_{ui} = I(r_{ui} > 0) \tag{4.1}$$

$$c_{ui} = 1 + \alpha \log\left(1 + \epsilon^{-1} r_{ui}\right) \tag{4.2}$$

where r_{ui} is the rating given by user u to item i. $I(x)$ is a indicator function. Variable p_{ui} indicates whether user u prefer item i and c_{ui} is a confidence variable.

Thus, the objective function of WMF can be written as:

$$\min_{x*,y*} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u \|x_u\|^2 \sum_i \|y_i\|^2) \tag{4.3}$$

4.2 Deep Convolutional Neural Network

Since CNN is a supervised deep model, authors use latent factors learned by WMF as ground truth to train a deep CNN with music signals as inputs. The paper introduces two objective functions, as Eq. 4.4 and Eq. 4.2, to train a CNN.

$$\min_\theta \|y_i - y_i'\|^2 \tag{4.4}$$

$$\min_\theta \sum_{u,i} c_{ui}(p_{ui} - x_u y_i')^2 \tag{4.5}$$

In Eq. 4.4, the CNN is trained to minimize the square error between the output vector of CNN(y_i') and item latent factors(y_i) learned by WMF. In Eq. 4.2, the objective function is used to train the CNN. θ denotes the parameters of the CNN.

4.3 Critique

This paper introduced the first deep model for music recommendation and demonstrated effective features can be learned from music signals via deep models. However, there are still several potential flaws.

First, the proposed model employs latent factors learned from WMF as ground truth to train a deep CNN. Although WMF is efficient to learn latent factors from implicit feedback, features approximated by WMF is still is not accurate enough to be used as ground truth to train CNN.

Meta data, such artists, genre or album, is not utilized. Although CNN can capture patterns existing in music signals, meta data is still informative and can be fused to learn item features.

Chapter 5

CoNN: Joint Modeling of Users and Items Using Reviews for Recommendation

To jointly model users and items using review texts for rating prediction problems, we propose Cooperative Neural Network (CoNN) to learn hidden latent features for users and items jointly using two coupled neural networks such that the rating prediction accuracy is maximized. One of the networks models user behavior using the reviews written by the user, and the other network models item properties using the written reviews for the item. The learned latent features for user and item are used to predict the corresponding rating in a layer introduced on the top of both networks. This interaction layer is motivated by matrix factorization techniques [1] to let latent factors of users and items interact with each other.

To capture semantic meaning existing in review texts, CoNN represents review texts using pre-trained word-embedding technique [47, 54, 55] to extract semantic information of the reviews. Recently, this representation has shown very good results in many Natural Language Processing (NLP) tasks [47, 48, 56, 57].

5.1 Architecture

The architecture of the proposed model for rating prediction is shown in Figure 5.1. The model consists of two parallel neural networks coupled in the last layer, one network for users (Net_u) and one network for items (Net_i). User reviews and item reviews are given to Net_u and Net_i respectively as inputs, and corresponding rating is produced as

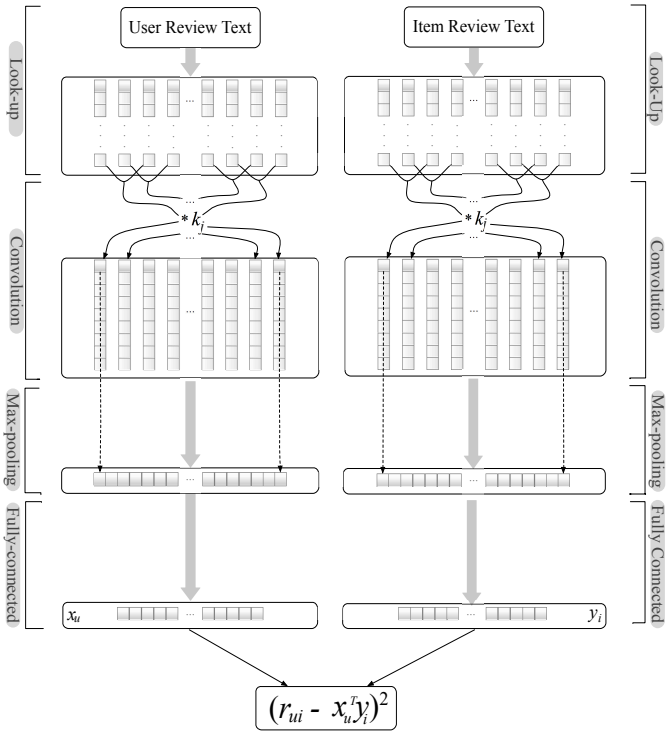

FIGURE 5.1: The architecture of CoNN

the output. In the first layer, denoted as look-up layer, review text for users or items are represented as matrices of word embeddings to capture the semantic information in the review texts. Next layers are the common layers used in CNN based models to discover multiple levels of features for users and items, including convolution layer, max pooling layer, and fully connected layer. Also, a top layer is added on the top of the two networks to let the hidden latent factors of user and item to interact with each other. This layer calculates an objective function that measures the rating prediction error using the latent factors produced by Net_u and Net_i. Note that Net_u and Net_i only differ in their inputs.

5.2 Network Training

$$J = \sum_{(u,i,r_{ui}) \in T} (r_{ui} - x_u^T y_i)^2 \tag{5.1}$$

Given a set of training set $\mathcal{T}$ consists of N tuples, we optimize the model through stochastic gradient descent over shuffled mini-batches. Our network is trained by minimizing Eq. 5.1. First, we take derivatives of J with respect to x_u and y_i, as shown in Eq. 5.2 and Eq. 5.3.

$$\frac{\partial J}{\partial x_u} = \sum_{(u,i,r_{ui}) \in T} -2(r_{ui} - x_u^T y_i) y_i \tag{5.2}$$

$$\frac{\partial J}{\partial y_i} = \sum_{(u,i,r_{ui})\in T} -2(r_{ui} - x_u^T y_i)x_u \tag{5.3}$$

Once we have estimated gradients with respect to x_u and y_i, we use backpropagation algorithm [25] to efficiently compute gradients of networks as the one proposed in [28, 56]. Because the parameters are in different layers of our networks, we apply the differentiation chain rule through the network until the first layer is reached.

Given the network with parameter set θ, mini-batch Stochastic Gradient Descent (SGD) [58] is used to minimize Eq. 5.1 with respect to θ. In each iteration, SGD randomly selects a batch of training examples (u, v, r) from the training set, computes its derivatives with respect to each parameter of the model and updates its parameters. The update rule is as following:

$$\theta \quad \leftarrow \theta - \lambda \frac{\partial J}{\partial \theta} \tag{5.4}$$

where λ is the learning rate.

5.3 Critique

Generally, Cooperative Neural Network (CoNN) is based on an observation that both user reviews and item reviews can reveal different aspects of users and items, this model couples two Neural Networks together. User reviews and item reviews are taken as inputs to discover user and item features. By introducing a shared layer on the top of these two NNs to couple them together, user and item representations are mapped into a common feature space and can effectively interact with each other to accurately model ratings.

One drawback of CoNN is that, similar to MF, CoNN can not deal with those users or items without ratings. Although experiments show that CoNN is effective to those users with a few ratings, CoNN is unable to handle users without a single rating. In the future, to enable CoNN produce reasonable recommendations to newly joining users, we can incorporate user information, such as ages or gender, into the model.

Chapter 6

Conclusion

In this article, we firstly introduce techniques involved in deep learning and recommender systems. Then, a survey and critique of deep learning on recommender systems are provided.

Although deep learning poses a great impact in various areas, deep learning techniques applied to recommender systems have not been fully exploited. Traditional recommender systems tend to recommend items based on ratings. However, due to the data sparsity problem, information existing in ratings is not sufficient to approximate latent factors of users and items. To alleviate the data sparsity problem, content-based recommender systems are introduced. Besides ratings, there are additional information, such as review texts, images or user profiles, can be utilized. Content-based recommender systems exploit these information to learn user preferences and item properties. However, most of content-based approaches are based on hand-crafted features for users and items. Although these features can be effective in different circumstances, hand-crafted features often require extensive human labor and often rely on expert knowledge. Also, usually, hand-crafted features are not generalized well. Deep learning techniques enable model to automatically learn features for users and items from different resources. These features are generalized well and can be effectively used to improve the quality of recommendation.

In the work of [9], they build a RBM model to capture user and item rating patterns. Although this model only utilize ratings and is not deep enough to capture complex features, it achieves better results by comparing to MF. It is the first attempt to apply deep learning techniques to recommender systems and show encouraging results. Later, due to the advantage of deep learning in modeling text, audio and images, contented-based deep recommender systems are introduced. By either modeling text or audio

signals, these deep models alleviate the sparsity problem and achieves state-of-the-art performance.

Overall, due to the limitation of the traditional recommendation approaches, the potential of content information has not been fully exploited. With the help of the advantage of deep learning in modeling different types of data, deep recommender systems can better understand what customers need and further improve recommendation quality.

Bibliography

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

[2] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192, 2002.

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

[4] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009:4, 2009.

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

[6] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237. ACM, 1999.

[7] Koji Miyahara and Michael J Pazzani. Collaborative filtering with the simple bayesian classifier. In *PRICAI 2000 Topics in Artificial Intelligence*, pages 679–689. Springer, 2000.

[8] Bin Shen, Xiaoyuan Su, Russell Greiner, Petr Musilek, and Corrine Cheng. Discriminative parameter learning of general bayesian network classifiers. In *Tools with Artificial Intelligence, 2003. Proceedings. 15th IEEE International Conference on*, pages 296–305. IEEE, 2003.

[9] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264. Curran Associates, Inc., 2007.

[10] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Multi-facet rating of product reviews. In *Advances in Information Retrieval*, pages 461–472. Springer, 2009.

[11] Yohan Jo and Alice H Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824. ACM, 2011.

[12] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[13] Ana-Maria Popescu and Orena Etzioni. Extracting product features and opinions from reviews. In *Natural language processing and text mining*, pages 9–28. Springer, 2007.

[14] Michael Gamon. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, page 841. Association for Computational Linguistics, 2004.

[15] Niklas Jakob, Stefan Hagen Weber, Mark Christoph Müller, and Iryna Gurevych. Beyond the stars: exploiting free-text user reviews to improve the accuracy of movie recommendations. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 57–64. ACM, 2009.

[16] Samuel Brody and Noemie Elhadad. An unsupervised aspect-sentiment model for online reviews. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 804–812. Association for Computational Linguistics, 2010.

[17] Ivan Titov and Ryan McDonald. Modeling online reviews with multi-grain topic models. In *Proceedings of the 17th international conference on World Wide Web*, pages 111–120. ACM, 2008.

[18] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[19] Guang Ling, Michael R Lyu, and Irwin King. Ratings meet reviews, a combined approach to recommend. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 105–112. ACM, 2014.

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

[21] Yao Wu and Martin Ester. FLAME: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *WSDM*, pages 199–208. ACM, 2015.

[22] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *KDD*, pages 193–202. ACM, 2014.

[23] Yang Bao, Hui Fang, and Jie Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, pages 2–8. AAAI Press, 2014.

[24] Geoffrey Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[25] Yves Chauvin and David E Rumelhart. *Backpropagation: theory, architectures, and applications*. Psychology Press, 1995.

[26] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29 (6):82–97, 2012.

[27] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[28] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[29] Yoshua Bengio. Learning deep architectures for ai. *Machine Learning*, 2(1):1–127, 2009.

[30] Yoshua Bengio and Yann LeCun. Scaling learning algorithms towards ai. *Large-scale kernel machines*, 34:1–41, 2007.

[31] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[32] Fu Jie Huang, Y-Lan Boureau, Yann LeCun, et al. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[33] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of ACL*, 2014.

[34] Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. Encoding source language with convolutional neural network for machine translation. *arXiv preprint arXiv:1503.01838*, 2015.

[35] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[36] Cicero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. Classifying relations by ranking with convolutional neural networks. *arXiv preprint arXiv:1504.06580*, 2015.

[37] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann

machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[38] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

[39] Xinxi Wang and Ye Wang. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*, pages 627–636. ACM, 2014.

[40] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[41] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[42] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. International World Wide Web Conferences Steering Committee, 2015.

[43] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1235–1244. ACM, 2015.

[44] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[45] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[46] Graham W Taylor, Geoffrey E Hinton, and Sam T Roweis. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, pages 1345–1352, 2006.

[47] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[48] Joseph Turian, Lev Ratinov, and Yoshua Bengio. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 384–394. Association for Computational Linguistics, 2010.

[49] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual*

international ACM SIGIR conference on Research and development in information retrieval, pages 253–260. ACM, 2002.

[50] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408, 2010.

[51] Hanna M Wallach. Topic modeling: beyond bag-of-words. In *Proceedings of the 23rd international conference on Machine learning*, pages 977–984. ACM, 2006.

[52] Malcolm Slaney. Web-scale multimedia analysis: Does content matter? *MultiMedia, IEEE*, 18(2):12–15, 2011.

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

[54] Joseph Turian, Lev Ratinov, Yoshua Bengio, and Dan Roth. A preliminary evaluation of word representations for named-entity recognition. In *NIPS Workshop on Grammar Induction, Representation of Language and Language Learning*, pages 1–8, 2009.

[55] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[56] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[57] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[58] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.