Joint Deep Modeling of Users and Items Using Reviews for Recommendation

Lei Zheng
Department of Computer
Science
University of Illinois at Chicago
Chicago, U.S.
lzheng21@uic.edu

Vahid Noroozi
Department of Computer
Science
University of Illinois at Chicago
Chicago, U.S.
vnoroo2@uic.edu

Philip S. Yu
Department of Computer
Science
University of Illinois at Chicago
Chicago, U.S.
psyu@uic.edu

ABSTRACT

A large amount of information exists in reviews written by users. This source of information has been ignored by most of the current recommender systems while it can potentially alleviate the sparsity problem and improve the quality of recommendations. In this paper, we present a deep model to learn item properties and user behaviors jointly from review text. The proposed model, named Deep Cooperative Neural Networks (DeepCoNN), consists of two parallel neural networks coupled in the last layers. One of the networks focuses on learning user behaviors exploiting reviews written by the user, and the other one learns item properties from the reviews written for the item. A shared layer is introduced on the top to couple these two networks together. The shared layer enables latent factors learned for users and items to interact with each other in a manner similar to factorization machine techniques. Experimental results demonstrate that DeepCoNN significantly outperforms all baseline recommender systems on a variety of datasets.

CCS Concepts

•**Information systems** → **Collaborative filtering; Recommender systems; **•**Computing methodologies** → *Neural networks;*

Keywords

Recommender Systems, Deep Learning, Convolutional Neural Networks, Rating Prediction

1. INTRODUCTION

The variety and number of products and services provided by companies have increased dramatically during the last decade. 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 help customers by presenting products or services that are likely of interest to them based on their preferences, needs, and past buying behaviors. Nowadays, many people use recommender systems in their daily life such as online shopping, reading articles, and watching movies.

Many of the prominent approaches employed in recommender systems [13] are based on Collaborative Filtering (CF) techniques. The basic idea of these techniques is that people who share similar preferences in the past tend to have similar choices in the future. Many of the most successful CF techniques are based on matrix factorization [13]. They find common factors that can be the underlying reasons for 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.

Although CF techniques have shown good performance for many applications, the sparsity problem is considered as one of their significant challenges [13]. The sparsity problem arises when the number of items rated by users is insignificant to the total number of items. It happens in many real applications. It is not easy for CF techniques to recommend items with few ratings or to give recommendations to users with few ratings.

One of the approaches employed to address this lack of data is using the information in review text [16, 17]. In many recommender systems, other than the numeric ratings, users can write reviews for the products. Users explain the reasons behind their ratings in text reviews. The reviews contain information which can be used to alleviate sparsity problem. One of the drawbacks of most current CF techniques is that they model users and items just based on numeric ratings provided by users and ignore the abundant information existed in the review text. Recently, some studies [17] [16] have shown that using review text can improve the prediction accuracy of recommender systems, in particular for items and users with few ratings [34].

In this paper, we propose a Neural Network (NN) based model, named Deep **Co**operative **N**eural **N**etworks (Deep-CoNN), to model users and items jointly using review text for rating prediction problems. The proposed model learns

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 [13] to let latent factors of users and items interact with each other.

To the best of our knowledge, DeepCoNN is the first deep model that represents both users and items in a joint manner using reviews. It makes the model scalable and also suitable for online learning scenarios where the model needs to get updated continuously with new data. Another key contribution is that DeepCoNN represents review text using pre-trained word-embedding technique [21, 20] to extract semantic information from the reviews. Recently, this representation has shown excellent results in many Natural Language Processing (NLP) tasks [7, 4, 21]. Moreover, a significant advantage of DeepCoNN compared to most other approaches [17, 16] which benefit from reviews is that it models users and items in a joint manner with respect to prediction accuracy. Most of the similar algorithms perform the modeling independently of the ratings. Therefore, there is no guarantee that the learned factors can be beneficial to the rating prediction.

The experiments on real-world datasets including *Yelp*, *Amazon* [19], and *Beer* [18] show that DeepCoNN outperforms all the compared baselines in prediction accuracy. Also, the proposed algorithm increases the performance for users and items with fewer ratings more than the ones with a higher number of ratings. It shows that DeepCoNN alleviates the sparsity problem by leveraging review text.

Our contributions and also advantages of DeepCoNN can be summarized as follows:

- The proposed Deep Cooperative Neural Networks (Deep-CoNN) jointly model user behaviors and item properties using text reviews. The extra shared layer at the top of two neural networks connects the two parallel networks such that user and item representations can interact with each other to predict ratings. To the best of our knowledge, DeepCoNN is the first one that jointly models both user and item from reviews using neural networks.

- It represents review text as word-embeddings using pre-trained deep models. The experimental results demonstrate that the semantic meaning and sentimental attitudes of reviews in this representation can increase the accuracy of rating prediction. All competing techniques which are based on topic modeling [38, 3, 8] use the traditional *bag of words* techniques.

- It does not only alleviate the problem of sparsity by leveraging reviews, but also improves the overall performance of the system significantly. It outperforms state-of-the-art techniques [17, 33, 25, 35] in terms of prediction accuracy on all of the evaluated datasets including Yelp, 21 categories of Amazon, and Beer (see Section 3).

The rest of the paper is organized as follows. In Section 2, we describe DeepCoNN in detail. Experiments are presented in Section 3 to analyze DeepCoNN and demonstrate its effectiveness compared to the state-of-the-art techniques for recommendation systems. In Section 4, we give a short review of the works related to our study. Finally, conclusions are presented in Section 5.

2. METHODOLOGY

The proposed model, DeepCoNN, is described in detail in this section. DeepCoNN models user behaviors and item properties using reviews. It learns hidden latent factors for users and items by exploiting review text such that the learned factors can estimate the ratings given by users. It is done with a CNN based model consisting of two parallel neural networks, coupled to each other with a shared layer at the top. The networks are trained in a joint manner to predict the ratings with minimum prediction error. We first describe notations used throughout this paper and formulate the definition of our problem. Then, the architecture of DeepCoNN and the objective function to get optimized is explained. Finally, we describe how to train this model.

2.1 Definition and Notation

A set of training set $\mathcal{T}$ consists of N tuples. Each tuple (u, i, r_{ui}, w_{ui}) denotes a review written by user u for item i with rating r_{ui} and text review of w_{ui}. The mathematical notations used in this paper are summarized in Table 1.

Table 1: **Notations**

Symbols	Definitions and Descriptions
$d^u_{1:n}$	user or item u's review text consisting of n words
$V^u_{1:n}$	word vectors of user or item u
w_{ui}	a review text written by user u for item i
o_j	the output of j_{th} neuron in the convolutional layer
n_i	the number of neurons in the layer i
K_j	the j_{th} kernel in the convolutional layer
b_j	the bias of j_{th} convolutional kernel
g	the bias of the fully connected layer
z_j	the j_{th} feature map in the convolutional layer
W	the weight matrix of the fully connected layer
t	the window size of convolutional kernel
c	the dimension of word embedding
$\mathbf{x_u}$	the output of Net_u
$\mathbf{y_i}$	the output of Net_i
λ	the learning rate

2.2 Architecture

The architecture of the proposed model for rating prediction is shown in Figure 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 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 text. 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 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. In the following subsections, since Net_u and Net_i only differ in their inputs, we focus on illustrating the process for Net_u in detail. The same process is applied for Net_i with similar layers.

2.3 Word Representation

A word embedding $f : M \to \Re^n$, where M represents the dictionary of words, is a parameterized function mapping words to n-dimensional distributed vectors. Recently, this approach has boosted the performance in many NLP applications [12, 7]. DeepCoNN uses this representation technique to exploit the semantics of reviews. In the look-up layer, reviews are represented as a matrix of word embeddings to extract their semantic information. To achieve it, all the reviews written by user u, denoted as user reviews, are merged into a single document $d_{1:n}^u$, consisting of n words in total. Then, a matrix of word vectors, denoted as $V_{1:n}^u$, is built for user u as follows:

$$V_{1:n}^u = \phi(d_1^u) \oplus \phi(d_2^u) \oplus \phi(d_3^u) \oplus \quad ... \quad \oplus \phi(d_n^u), \quad (1)$$

where d_k^u indicates the kth word of document $d_{1:n}^u$, look-up function $\phi(d_k^u)$ returns the corresponding c-dimensional word vector for the word d_k^u, and $\oplus$ is the concatenation operator. It should be considered that the order of words is preserved in matrix $V_{1:n}^u$ that is another advantage of this representation comparing to *bag-of-words* techniques.

2.4 CNN Layers

Next layers including convolution layer, max pooling, and fully connected layer follow the CNN model introduced in [7]. Convolution layer consists of m neurons which produce new features by applying convolution operator on word vectors $V_{1:n}^u$ of user u. Each neuron j in the convolutional layer uses filter $K_j \in \Re^{c \times t}$ on a window of words with size t. For $V_{1:n}^u$, we perform a convolution operation regarding each kernel K_j in the convolutional layer.

$$z_j = f(V_{1:n}^u * K_j + b_j) \quad (2)$$

Here symbol $*$ is convolution operator, b_j is a bias term and f is an activation function. In the proposed model, we use Rectified Linear Units (ReLUs) [22]. It is defined as Eq. 3. Deep convolutional neural networks with ReLUs train several times faster than their equivalents with tanh units [14].

$$f(x) = max\{0, x\} \quad (3)$$

Following the work of [7], we then apply Eq. 4, a max pooling operation, over the feature map and take the maximum value as the feature corresponding to this particular kernel. The most important feature of each feature map, which has the highest value, has been captured. This pooling scheme can naturally deal with the varied length of the text. After the max pooling operation, convolutional results are reduced to a fixed size vector.

$$o_j = max\{z_1, z_2, ..., z_{(n-t+1)}\} \quad (4)$$

We have described the process by which one feature is extracted from one kernel. The model uses multiple filters to obtain various features and the output vector of the convolutional layer is as Eq. 5.

$$O = \{o_1, o_2, o_3, ..., o_{n_1}\}, \quad (5)$$

where n_1 denotes the number of kernel in the convolutional layer.

$$\mathbf{x_u} = f(W \times O + g) \quad (6)$$

The results from the max-pooling layer are passed to a fully connected layer with weight matrix W. As shown in Eq. 6, the output of the fully connected layer $\mathbf{x_u} \in \Re^{n_2 \times 1}$ is considered as features for user u. Finally, the outputs of both user and item CNN $\mathbf{x_u}$ and $\mathbf{y_i}$ can be obtained.

2.5 The Shared Layer

Although these outputs can be viewed as features of users and items, they can be in different feature space and not comparable. Thus, to map them into the same feature space, we introduce a shared layer on the top to couple Net_u and Net_i. First, let us concatenate $\mathbf{x_u}$ and $\mathbf{y_i}$ into a single vector $\hat{\mathbf{z}} = (\mathbf{x_u}, \mathbf{y_i})$. To model all nested variable interactions in $\hat{\mathbf{z}}$, we introduce Factorization Machine (FM) [24] as the estimator of the corresponding rating. Therefore, given a batch of N training examples $\mathcal{T}$, we can write down its cost as Eq. 7.

$$J = \hat{w}_0 + \sum_{i=1}^{|\hat{z}|} \hat{w}_i \hat{z}_i + \sum_{i=1}^{|\hat{z}|} \sum_{j=i+1}^{|\hat{z}|} \langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle \hat{z}_i \hat{z}_j, \quad (7)$$

where $\hat{w}_0$ is the global bias, $\hat{w}_i$ models the strength of the i_{th} variable in $\hat{z}$ and $\langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle = \sum_{f=1}^{|\hat{z}|} \hat{\mathbf{v}}_{i,f} \hat{\mathbf{v}}_{j,f}$. $\langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle$ models the second order interactions.

2.6 Network Training

Our network is trained by minimizing Eq. 7. We take derivatives of J with respect to z, as shown in Eq. 8.

$$\frac{\partial J}{\partial \hat{z}_i} = \hat{w}_i + \sum_{j=i+1}^{|\hat{z}|} \langle \hat{\mathbf{v}}_i, \hat{\mathbf{v}}_j \rangle \hat{z}_j \quad (8)$$

The derivatives of other parameters in different layers can be computed by applying differentiation chain rule.

Given a set of training set $\mathcal{T}$ consisting of N tuples, we optimize the model through RMSprop [30] over shuffled mini-batches. RMSprop is an adaptive version of gradient descent which adaptively controls the step size with respect to the absolute value of the gradient. It does it by scaling the update value of each weight by a running average of its gradient norm. The updating rules for parameter set θ of the networks are as the following:

$$r_t \leftarrow 0.9(\frac{\partial J}{\partial \theta})^2 + 0.1 r_{t-1} \quad (9)$$

$$\theta \leftarrow \theta - (\frac{\lambda}{\sqrt{r_t} + \epsilon}) \frac{\partial J}{\partial \theta}, \quad (10)$$

where λ is the learning rate, ϵ is a small value added for numerical stability. Additionally, to prevent overfitting, the dropout [28] strategy has also been applied to the fully connected layers of the two networks.

2.7 Some Analysis on DeepCoNN

2.7.1 Word Order Preservation

Most of the recommender systems which use reviews in the modeling process employ topic modeling techniques to model users or items [6]. Topic modeling techniques infer
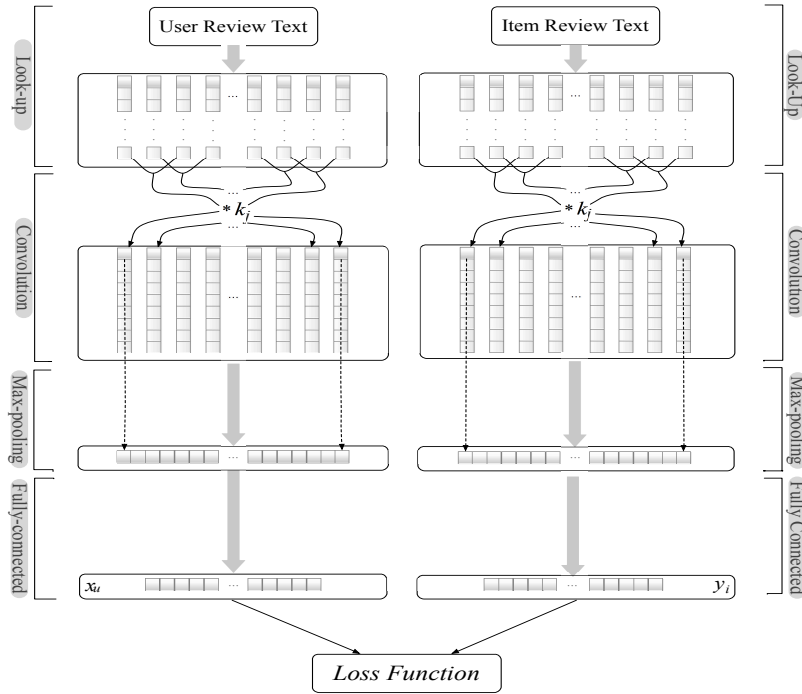
Figure 1: The architecture of the proposed model

latent topic variables using the *bag-of-words* assumption, in which word order is ignored. However, in many text modeling applications, word order is crucial [32]. DeepCoNN is not based on topic modeling and uses word embeddings to create a matrix of word vectors where the order of words are preserved. In this way, convolution operations make use of the internal structure of data and provide a mechanism for efficient use of words' order in text modeling [11].

2.7.2 *Online Learning*

Scalability and handling dynamic pools of items and users are considered as critical needs of many recommender systems. The time sensitivity of recommender systems poses a challenge in learning latent factors in an online fashion. DeepCoNN is scalable to the size of the training data, and also it can easily get trained and updated with new data because it is based on NN. Updating latent factors of items or users can get performed independently from historical data. All the approaches which employ topic modeling techniques do not benefit from these advantages to this extent.

3. EXPERIMENTS

We have performed extensive experiments on a variety of datasets to demonstrate the effectiveness of DeepCoNN compared to other state-of-the-art recommender systems. We first present the datasets and the evaluation metric used in our experiments in Section 3.1. The baseline algorithms selected for comparisons are explained in Section 3.2. Experimental settings are given in Section 3.3. Performance evaluation and some analysis of the model are discussed in sections 3.4 and 3.5 respectively.

3.1 Datasets and Evaluation Metric

In our experiments, we have selected the following three datasets to evaluate our model.

- **Yelp**: It is a large-scale dataset consisting of restaurant reviews, introduced in the 6th round of *Yelp Challenge* [1] in 2015. It contains more than 1M reviews and ratings.

- **Amazon**: Amazon Review dataset [19] contains product reviews and metadata from Amazon website[2]. It includes more than 143.7 million reviews spanning from May 1996 to July 2014. It has 21 categories of items, and as far as we know, this is the largest public available rating dataset with text reviews.

- **Beer**: It is a beer review dataset extracted from *ratebeer.com*. The data span a period of more than 10 years, including almost 3 million reviews up to November 2011 [18].

As we can see in Table 2, all datasets contain more than half a million of reviews. However, in *Yelp* and *Amazon*, customers provide less than six pair of reviews and ratings on average which shows these two datasets are extremely sparse. This sparsity can largely deteriorate the performance of recommender systems. Besides, in all datasets, each review consists of less than 150 words on average.

In our experiments, we adopt the well-known Mean Square Error (MSE) to evaluate the performance of the algorithms. It is selected because most of the related works have used the same evaluation metric[17, 16, 1]. MSE can be defined as follows:

$$MSE = \frac{1}{N} \sum_{n=1}^{N} (r_n - \hat{r}_n)^2, \qquad (11)$$

where r_n is the nth observed value, $\hat{r}_n$ is the nth predicted value and N is the total number of observations.

Table 2: **The Statistics of the datasets**

Class	#users	#items	#review	#words	#reviews per user	#words per review
Yelp	366,715	60,785	1,569,264	198M	4.3	126.41
Amazon	6,643,669	2,441,053	34,686,770	4.053B	5.2	116.67
Beer	40,213	110,419	2,924,127	154M	72.7	52.67

3.2 Baselines

To validate the effectiveness of DeepCoNN, we have selected three categories of algorithms for evaluations: (i) purely rating based models. We chose Matrix Factorization (MF) and Probabilistic Matrix Factorization (PMF) to validate that review information is helpful for recommender systems, (ii) topic modeling based models which use review information. Most of the recommender systems which take reviews into consideration are based on topic modeling techniques. To compare our model with topic modeling based recommender systems, we select three representative models: Latent Dirichlet Allocation (LDA) [5], Collaborative Topic Regression (CTR) [33] and Hidden Factor as Topic (HFT) [17], and (iii) deep recommender systems. In [35], authors have proposed a state-of-the-art deep recommender system named Collaborative Deep Learning (CDL). Note that all the baselines except MF and PMF have incorporated review information into their models to improve prediction.

- **MF**: **M**atrix **F**actorization [13] is the most popular CF-based recommendation method. It only uses rating matrix as input and estimates two low-rank matrices to predict ratings. In our implementation, Alternating Least Squares (ALS) technique is adopted to minimize its objective function.

- **PMF**: **P**robabilistic **M**atrix **F**actorization is introduced in [25]. It models latent factors of users and items by Gaussian distributions.

- **LDA**: **L**atent **D**irichlet **A**llocation is a well-known topic modeling algorithm presented in [5]. In [17], it is proposed to employ LDA to learn a topic distribution from a set of reviews for each item. By treating the learned topic distributions as latent features for each item, latent features for each user is estimated by optimizing rating prediction accuracy with gradient descent.

- **CTR**: **C**ollaborative **T**opic **R**egression has been proposed by [33]. It showed very good performance on recommending articles in a one-class collaborative filtering problem where a user is either interested or not.

- **HFT**: **H**idden **F**actor as **T**opic proposed in [17] employs topic distributions to learn latent factors from user or item reviews. The authors have shown that item specific topic distributions produce more accurate predictions than user specific ones. Thus, we report the results of HFT learning from item reviews.

- **CDL**: **C**ollaborative **D**eep **L**earning tightly couples a Bayesian formulation of the stacked denoising auto-encoders and PMF. The middle layer of auto-encoders serves as a bridge between auto-encoders and PMF.

3.3 Experimental Settings

We divided each dataset shown in Table 2 into three sets of training set, validation set, and test set. We use 80% of each
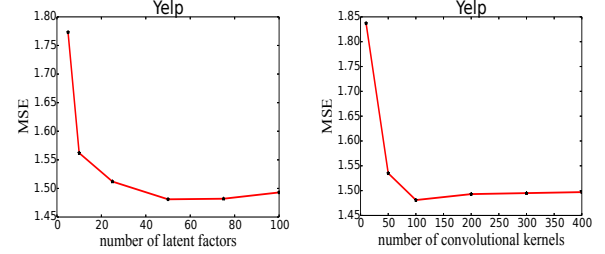

Figure 2: The impact of the number of latent factors and convolutional kernels on the performance of DeepCoNN in terms of MSE (*Yelp* Dataset).

dataset as the training set, 10% is treated as the validation set to tune the hyper-parameters, and the rest is used as the test set. All the hyper-parameters of the baselines and DeepCoNN are selected based on the performance on the validation set.

For MF and PMF, we used grid search to find the best values for the number of latent factors from $\{25, 50, 100, 150, 200\}$, and regularization parameter from $\{0.001, 0.01, 0.1, 1.0\}$.

For LDA, CTR and HFT, the number of topics K is selected from $\{5, 10, 20, 50, 100\}$ using the validation set. We set $K = 10$ for LDA and CTR. The CTR model solves the *one-class collaborative filtering problem* [23] by using two different values for the precision parameter c of a Gaussian distribution. Following the work of [16], in our experiments, we set precision c as the same for all the observed ratings for rating prediction. HFT-k ($k = 10, 50$) are included to show the impact of the number of latent factors for HFT. By performing a grid search on the validation set, we set hyper-parameters $\alpha = 0.1$, $\lambda_u = 0.02$ and $\lambda_v = 10$ for CTR and HFT. To optimize the performance of CDL, we performed a grid search on the hyper-parameters λ_u, λ_v, λ_n, λ_w and L. Similar with CTR, the confidence parameter c_{ij} of CDL is set as the same for all observed ratings.

We empirically studied the effects of two important parameters of DeepCoNN: the number of latent factors($|\mathbf{x_u}|$ and $|\mathbf{y_i}|$) and the number of convolutional kernels: n_1. In Figure 2, we show the performance of DeepCoNN on the validation set of *Yelp* with varying $|\mathbf{x_u}|$ and $|\mathbf{y_i}|$ from 5 to 100 and n_1 from 10 to 400 to investigate its sensitivity. As it can be seen, it does not improve the performance when the number of latent factors and number of kernels is greater than 50 and 100 respectively. Thus, we set $|\mathbf{x_u}| = |\mathbf{y_i}| = 50$ and $n_1 = 100$. Other hyper-parameters: t, c, λ and batch size are set as 3, 300, 0.002 and 100, respectively. These values were chosen through a grid search on the validation sets. We used a pre-trained word embeddings which are trained on more than 100 billion words from Google News [21] [3].

Our models are implemented in *Theano* [29], a well-known Python library for machine learning and deep learning. The

---

[3]https://code.google.com/archive/p/word2vec/

NVIDIA CUDA Deep Neural Network4 (cuDNN v4) accelerated our training process. All models are trained and tested on an NVIDIA Tesla K40 GPU.

3.4 Performance Evaluation

The performance of DeepCoNN and the baselines (see Section 3.2) are reported in terms of MSE in Tables 3. Table 3 shows the results on the three datasets including the performance averaged on all 21 categories of *Amazon*. The experiments are repeated 3 times, and the averages are reported with the best performance shown in bold. The last column indicates the percentage of improvements gained by DeepCoNN compared to the best baseline in the corresponding category.

In Table 3, all models perform better on *Beer* dataset than on *Yelp* and *Amazon*. It is mainly related to the sparsity of *Yelp* and *Amazon*. Although PMF performs better than MF on *Yelp*, *Beer*, and most categories of *Amazon*, both techniques do not show good performance compared to the ones which use reviews. It validates our hypothesis that review text provides additional information, and considering reviews in models can improve rating prediction.

Although simply employing LDA to learn features from item reviews can help the model to achieve improvements, LDA models reviews independent of ratings. Therefore, there is no guarantee that the learned features can be beneficial to rating prediction. Therefore, by modeling ratings and reviews together, CTR and HFT attain additional improvements. Among those topic modeling based models (LDA, CTR and HFT), both HFT-10 and HFT-50 perform better in all three datasets.

With the capability of extracting deep effective features from item review text, as we can see in Table 3, CDL outperforms all topic modeling based recommender systems and advances the state-of-the-art. However, in benefiting from joint modeling capacity and semantic meaning existing from review text, DeepCoNN beats the best baseline in *Yelp*, *Beer* and *Amazon* and gains **8.3%** improvement on average.

3.5 Model Analysis

Are the two parallel networks really cooperate to learn effective features from reviews? Does the proposed model benefit from the use of word embedding to exploit the semantic information in the review text? How much does the shared layer help in improving the predcition accuracy comparing to a simpler coupling approach? To answer these questions, we compare the DeepCoNN with its five variants: DeepCoNN-User, DeepCoNN-Item, DeepCoNN-TFIDF, DeepCoNN-Random and DeepCoNN-DP. These five variants are summarized in the following:

- **DeepCoNN-User**: The Net_i of DeepCoNN is substituted with a matrix. Each row of the matrix is the latent factors of one item. This matrix is randomly initialized and optimized during the training.

- **DeepCoNN-Item**: Similar with DeepCoNN-User, the Net_u of DeepCoNN is replaced with a matrix. Each row of the matrix is the latent factors of one user. This matrix is randomly initialized and optimized during the training.

- **DeepCoNN-TFIDF**: Instead of using word embedding, the TFIDF scheme is employed to represent review text as input to DeepCoNN.

- **DeepCoNN-Random**: Our baseline model where all word representations are randomly initialized as fixed-length vectors.

- **DeepCoNN-DP**: The factorization machine in the objective function is substitued with a simple dot product of $\mathbf{x_u}$ and $\mathbf{y_i}$.

The performance of DeepCoNN and its variants on *Yelp*, *Beer* and one category of the *Amazon* dataset: *Music Instruments* are given in Table 4.

To demonstrate that the two deep CNNs can cooperate with each other to learn effective latent factors from user and item reviews, DeepCoNN-User and DeepCoNN-Item are trained with only one CNN with review text as input and the other CNN is substituted with a list of latent variables as the parameters to get learned. In this manner, latent factors of users or items are learned without considering their corresponding review text. As it can be seen in Table 4, while DeepCoNN-User and DeepCoNN-Item achieve similar results, DeepCoNN delivers the best performance by modeling both users and items. It verifies that review text is necessary for modeling latent factors of both users and items. Also, it shows that review text has informative information that can help to improve the performance of recommendation.

Furthermore, to validate the effectiveness of word representation, we compare DeepCoNN with DeepCoNN-TFIDF and DeepCoNN-Random. The DeepCoNN-TFIDF and DeepCoNN-Random are trained to show that word embedding is helpful to capture semantic meaning existed in the review text. While the performance of DeepCoNN-TFIDF is slightly better than DeepCoNN-Random, they both perform considerably weaker than DeepCoNN. It shows the effectiveness of representing review text in semantic space for modeling the latent factors of items or users.

At last, to investigate the efficiency of the shared layer, DeepCoNN-DP is introduced that couples the two networks with a simpler objective function. The comparison shows the superiority of the factorization machine coupling. It can be the result of not only modeling the first order interactions but also the second order interactions between $\mathbf{x_u}$ and $\mathbf{y_i}$.

3.6 The Impact of the Number of Reviews

The cold start problem [27] is prevalent in recommender systems. In particular, when a new user joins or a new item is added to the system, their available ratings are limited. It would not be easy for the system to learn preferences of such users just from their ratings. It has been shown in some of the previous works that exploiting review text can help to alleviate this problem especially for users or items with few ratings [17]. In this section, we conduct a set of experiments to answer the following questions. Can DeepCoNN help to tackle the cold start problem? What is the impact of the number of reviews on the effectiveness of the proposed algorithm?

In Fig. 3, we have illustrated the reductions in MSE resulted from DeepCoNN compared to MF technique on three datasets of Yelp, Beer, and a group of Amazon (Music Instruments). By reduction in MSE, we mean the difference between the MSE of MF and the MSE of DeepCoNN. Users and items are categorized based on the number of their reviews, and reductions are plotted for both users and items groups. It can be seen that in all three datasets, reductions

Table 3: **MSE Comparison with baselines**. Best results are indicated in bold.

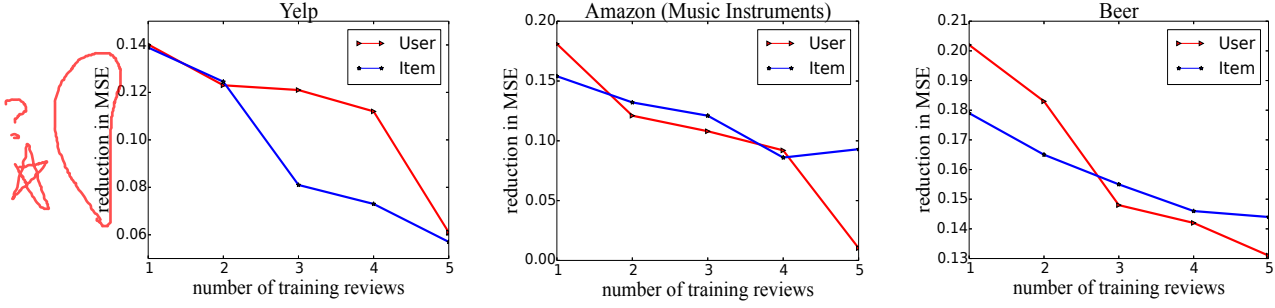| Dataset	MF	PMF	LDA	CTR	HFT-10	HFT-50	CDL	DeepCoNN	Improvement of DeepCoNN (%)
Yelp	1.792	1.783	1.788	1.612	1.583	1.587	1.574	**1.441**	8.5%
Amazon	1.471	1.460	1.459	1.418	1.378	1.383	1.372	**1.268**	7.6%
Beer	0.612	0.527	0.306	0.305	0.303	0.302	0.299	**0.273**	8.7%
Average on all datasets	**1.292**	**1.256**	**1.184**	**1.112**	**1.088**	**1.09**	**1.081**	**0.994**	8.3%

Figure 3: MSE improvement achieved by DeepCoNN compared to MF. For users and items with different number of training reviews, DeepCoNN gains different MSE reductions.

Table 4: **Comparing variants of the proposed model**. Best results are indicated in bold.

Model	Yelp	Amazon Music Instruments	Beer
DeepCoNN-User	1.577	1.373	0.292
DeepCoNN-Item	1.578	1.372	0.296
DeepCoNN-TFIDF	1.713	1.469	0.589
DeepCoNN-Random	1.799	1.517	0.627
DeepCoNN-DP	1.491	1.253	0.278
DeepCoNN	**1.441**	**1.233**	**0.273**

are positive, and DeepCoNN can achieve RMS reduction on all groups of users and items with few number of ratings. A more important advantage of DeepCoNN is that higher reductions are gained for groups with fewer ratings. It shows that DeepCoNN can alleviate the sparsity problem and help on the cold start problem.

It can also be seen that there exists a relation between the effectiveness of DeepCoNN and the number of ratings for a user or item. For users or items with a lower number of ratings, DeepCoNN reduction in MSE is higher. It shows that review text can be valuable information especially when we have limited information on the users or items.

4. RELATED WORKS

There are two categories of studies related to our work: techniques that model users and/or items by exploiting the information in online review text, and deep learning techniques employed for recommender systems. In this section, we give a short review of these two research areas and distinguish our work from the existing approaches.

The first studies that used online review text in rating prediction tasks were mostly focused on predicting ratings for an existing review [2, 38], while in our paper, we predict

the ratings from the history of review text written by a user to recommend desirable products to that user.

One of the pioneer works that explored using reviews to improve the rating prediction is presented in [10]. It found that reviews are usually related to different aspects, e.g., price, service, positive or negative feelings, that can be exploited for rating prediction. In [17], the authors proposed Hidden Factors as Topics (HFT) to employ topic modeling techniques to discover latent aspects from either item or user reviews. This method achieves significant improvement compared to models which only use ratings or reviews. A similar approach is followed in [3] with the main difference that it models user's and items' reviews simultaneously. In [8], a probabilistic model is proposed based on collaborative filtering and topic modeling. It uncovers aspects and sentiments of users and items, but it does not incorporate ratings during modeling reviews. Ratings Meet Reviews (RMR) [16] also tries to harness the information of both ratings and reviews. One difference between HFT and RMR is that RMR applies topic modeling techniques on item review text and aligns the topics with the rating dimensions to improve prediction accuracy.

Overall, one limitation of the above studies is that their textual similarity is solely based on lexical similarity. The vocabulary in English is very diverse, and two reviews can be semantically similar even with low lexical overlapping. The semantic meaning is of particular importance and has been ignored in these works. Additionally, reviews are represented by using *bag-of-words*, and words' order exists in reviews has not been preserved. At last, the approaches which employ topic modeling techniques suffer from a scalability problem and also cannot deal with new coming users and items.

Recently, several studies have been done to use neural network based models including deep learning techniques for recommendation tasks. Several works [26, 37, 15] model users and/or items from the rating matrix using neural net-

works like denoising auto-encoders or Restricted Boltzmann Machines (RBM). They are considered as collaborative based techniques because they just utilize the rating matrix and ignore review text unlike our approach.

In [31] and [36], deep models of CNN and Deep Belief Network (DBN) are introduced to learn latent factors from music data for music recommendation. In both models, initially, they find user and item latent factors using matrix factorization techniques. Then, they train a deep model such that it can reconstruct these latent factors for the items from the music content. A similar approach is followed in [35] for movie recommendation by using a generalized Stacked Auto Encoder (SAE) model. In all these works [31, 36, 35], an item's latent factors are learned from item's content and review text is ignored.

In [9], a multi-view deep model is built to learn the user and item latent factors in a joint manner and map them to a common space. The general architecture of the model seems to have some similarities to our proposed model, but it differs from ours in some aspects. Their model is a content-based recommender system and does not use review text. Moreover, their outputs are coupled with a cosine similarity objective function to produce latent factors with high similarity. In this way, user and item factors are not learned explicitly in relation to the rating information, and there is no guarantee that the learned factors can help the recommendation task.

All the above NN based approaches differ from DeepCoNN because they ignore review text. To the best of our knowledge, the only work which has utilized deep learning techniques to use review text to improve recommendation is presented in [1]. To use the information exists in reviews, they proposed a model consisting of a matrix factorization technique and a Recurrent Neural Network (RNN). The matrix factorization is responsible for learning the latent factors of users and items, and the RNN models the likelihood of a review using the item's latent factors. The RNN model is combined with the MF simply via a trade-off term as some sort of a regularization term to tame the curse of data sparsity. Due to the matrix factorization technique, handling new users and items is not trivial in this model unlike Deep-CoNN that handles them easily. Their proposed algorithm does not model users and items explicitly in a joint manner from their reviews, and it just uses reviews to regularize their model. In addition, since item text is represented by using *bag-of-words*, semantic meaning existing in words has not been explored.

5. CONCLUSION

It is shown that reviews written by users can reveal some info on the customer buying and rating behavior, and also reviews written for items may contain info on their features and properties. In this paper, we presented Deep Cooperative Neural Networks (DeepCoNN) which exploits the information exists in the reviews for recommender systems. DeepCoNN consists of two deep neural networks coupled together by a shared common layer to model users and items from the reviews. It makes the user and item representations mapped into a common feature space. Similar to MF techniques, user and item latent factors can effectively interact with each other to predict the corresponding rating.

In comparison with state-of-the-art baselines, DeepCoNN achieved **8.5%** and **7.6%** improvements on datasets of *Yelp*

and *Beer*, respectively. On *Amazon*, it outperformed all the baselines and gained **8.7%** improvement on average. Overall, **8.3%** improvement is attained by the proposed model on all three datasets.

Additionally, in the experiments by limiting modeling to just one of the users and items, we demonstrated that the two networks could not only separately learn user and item latent factors from review text but also cooperate with each other to boost the performance of rating prediction. Furthermore, we showed that word embedding could be helpful to capture semantic meaning of review text by comparing it with a variant of DeepCoNN which uses random or TF-IDF representations for reviews.

At last, we conducted experiments to investigate the impact of the number of reviews. Experimental results showed that for the users and items with few reviews or ratings, DeepCoNN obtains more reduction in MSE than MF. Especially, when only one review is available, DeepCoNN gains the greatest MSE reduction. Thus, it validates that Deep-CoNN can eïňĂectively alleviate the sparsity problem.

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] A. Almahairi, K. Kastner, K. Cho, and A. Courville. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154. ACM, 2015.

[2] S. Baccianella, A. Esuli, and F. Sebastiani. Multi-facet rating of product reviews. In *Advances in Information Retrieval*, pages 461–472. Springer, 2009.

[3] Y. Bao, H. Fang, and J. Zhang. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI*, pages 2–8. AAAI Press, 2014.

[4] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[6] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction*, 25(2):99–154, 2015.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[8] Q. Diao, M. Qiu, C. Wu, A. J. Smola, J. Jiang, and C. Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *KDD*, pages 193–202. ACM, 2014.

[9] A. M. Elkahky, Y. Song, and X. 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.

[10] N. Jakob, S. H. Weber, M. C. Müller, and I. 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.

[11] R. Johnson and T. Zhang. Effective use of word order for text categorization with convolutional neural networks. In *HLT-NAACL*, pages 103–112. The Association for Computational Linguistics, 2015.

[12] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[13] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[15] S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 811–820. ACM, 2015.

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

[17] J. McAuley and J. 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.

[18] J. McAuley, J. Leskovec, and D. Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 1020–1025. IEEE, 2012.

[19] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.

[20] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048, 2010.

[21] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[22] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[23] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[24] S. Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(3):57, 2012.

[25] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264. Curran Associates, Inc., 2007.

[26] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[27] A. I. Schein, A. Popescul, L. H. Ungar, and D. 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.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[29] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[30] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4:2, 2012.

[31] A. Van den Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

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

[33] C. Wang and D. 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.

[34] H. Wang, Y. Lu, and C. Zhai. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792. ACM, 2010.

[35] H. Wang, N. Wang, and D.-Y. 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.

[36] X. Wang and Y. 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.

[37] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems.

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