Neural Attentional Rating Regression with Review-level Explanations

Chong Chen
DCST, Tsinghua University
Beijing, China
cstchenc@163.com

Min Zhang*
DCST, Tsinghua University
Beijing, China
z-m@tsinghua.edu.cn

Yiqun Liu
DCST, Tsinghua University
Beijing, China
yiqunliu@tsinghua.edu.cn

Shaoping Ma
DCST, Tsinghua University
Beijing, China
msp@tsinghua.edu.cn

ABSTRACT

Reviews information is dominant for users to make online purchasing decisions in e-commerces. However, the usefulness of reviews is varied. We argue that less-useful reviews hurt model's performance, and are also less meaningful for user's reference. While some existing models utilize reviews for improving the performance of recommender systems, few of them consider the usefulness of reviews for recommendation quality. In this paper, we introduce a novel attention mechanism to explore the usefulness of reviews, and propose a Neural Attentional Regression model with Review-level Explanations (NARRE) for recommendation. Specifically, NARRE can not only predict precise ratings, but also learn the usefulness of each review simultaneously. Therefore, the highly-useful reviews are obtained which provide review-level explanations to help users make better and faster decisions. Extensive experiments on benchmark datasets of Amazon and Yelp on different domains show that the proposed NARRE model consistently outperforms the state-of-the-art recommendation approaches, including PMF, NMF, SVD++, HFT, and DeepCoNN in terms of rating prediction, by the proposed attention model that takes review usefulness into consideration. Furthermore, the selected reviews are shown to be effective when taking existing review-usefulness ratings in the system as ground truth. Besides, crowd-sourcing based evaluations reveal that in most cases, NARRE achieves equal or even better performances than system's usefulness rating method in selecting reviews. And it is flexible to offer great help on the dominant cases in real e-commerce scenarios when the ratings on review-usefulness are not available in the system.

CCS CONCEPTS

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

KEYWORDS

Recommender Systems, Neural Attention Network, Explainable Recommendation, Review Usefulness

WWW 2018, April 23–27, 2018, Lyons, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186070

Figure 1: Examples of less-useful (A, B) and highly-useful (C) reviews selected from the film "Iron Man" on Amazon. Review A just contains the rough preference of the consumer. Review B is talking about something else, but not about the film. In contrast, review C provides detailed information, which is more helpful for user's potential consumption.

ACM Reference Format:
Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *The Web Conference 2018, April 23–27, 2018, Lyons, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186070

1 INTRODUCTION

With the growing number of choices available online, recommender systems are playing a more and more important role in alleviating information overload, and have been widely adopted by many websites and applications. Collaborative Filtering(CF) is a dominant state-of-art recommendation methodology, which focus on the proper modeling of user preferences and item features by historical records such as ratings, clicks, and consumptions[20, 21, 24, 29]. Although CF techniques have shown good performance, they encounter an important problem in practical applications: can not provide explicit explanations about why an item is recommended. In recent years, some researchers have found that explanations in recommendation systems could be very beneficial[11, 44]. By

*Corresponding author

explaining how the system works, the system becomes more transparent and has the potential to increase users' confidence or trust in the system and help users make better (effectiveness) and faster (efficiency) decisions[44]. Lack of explainability weakens the ability to persuade users and help users on consumptions in real life[40].

In most e-commerce and review service websites like Amazon and Yelp, users are allowed to write free-text reviews along with a numerical star rating. The text reviews usually contain rich information about the item's features (e.g. quality, material, and color), and sometimes instructive suggestions, which are of great reference values for those who are going to make purchasing decisions.

However, it is difficult for users to get useful information from an immense number of available reviews, as the usefulness of them are varied. In this paper, the ***usefulness*** of a review is defined as whether it can provide detailed information about the item and help users make their purchasing decisions easily. In Figure 1, we show the examples of less-useful (A, B) reviews and highly-useful (C) review selected from the film "Iron Man" on Amazon. As we can see, compared with review C, review A only contains the rough opinion of the consumer, but shows no characteristic of the film, and review B has less relevance to the film and is somehow biased. Review C is also marked as helpful by 8 users in the system, which is called **"Rated_Useful"** in this paper. Less-useful reviews not only introduce noises which undermine the performance of recommender systems, but also are less useful to users.

Existing models integrate user reviews to enhance the performance of latent factor modeling[3, 25–27, 39, 46] and generate explanations[11, 32, 44]. Although they have achieved good results, they still have some inherent limitations. First, they do not consider the contributions of each review to item modeling, as well as the usefulness to other users. Second, their explanations are simple extractions of words or phrases from the reviews, which may twist the meaning of the original sentences[34]. To the best of our knowledge, we are the first to consider the usefulness of reviews for improving the performance and explainability of recommendation. We aim at developing a model that is capable of conducting rating prediction, and more importantly, it is able to pick out valuable reviews from the messy data simultaneously. Based on this work, the review-level explanations on whether and why an item is worth recommending can be provided.

To learn the usefulness of the reviews, we propose a Neural Attentional Regression model with Review-level Explanations (NARRE) in this paper, which utilizes the recent advance in neural network modelling – the attention mechanism to automatically assign weights to reviews in a distant supervised manner[1, 4]. Specifically, we propose a weighting function which is a multi-layer neural network and takes the characteristics of both users and items, as well as the content of reviews as input. What's more, inspired by[46], NARRE learns hidden latent features for users and items jointly using two parallel neural networks. One of the networks models user preferences using the reviews written by the user, and the other network models item features using the written reviews for the item. In the last layer, we draw on the Latent Factor Model[21] and extend it to a neural network for rating prediction. We evaluate NARRE extensively on four real-world datasets. Experimental results show that our model consistently outperforms the state-of-the-art methods including PMF[29], NMF[24], SVD++[20], HFT[27], and DeepCoNN[46].

The main contributions of this work are summarized as follows.

(1) We propose a novel idea that different reviews have different contributions on item modelling and lead to different usefulness to other users on consumptions.
(2) To the best of our knowledge, we are the first to introduce neural attention mechanism to build the recommendation model and select highly-useful reviews simultaneously, which helps to improve the performance and explainability of the recommender system.
(3) Experimental results on benchmark datasets show that our model achieves better rating prediction results than the state-of-the-art models, including matrix factorization based approaches and deep learning based DeepCoNN. Furthermore, crowd-sourcing based analyses on review usefulness have been made, which show that our selected reviews are equally useful or even better than the original users' usefulness-rated ones in the system. And it is flexible to offer great help on the dominant cases in real scenarios when the ratings on review-usefulness are not available in the system.

2 RELATED WORK

In recent years, matrix factorization (MF) has become the most popular collaborative filtering approach[35, 38]. The original MF model[21] was designed to model users' explicit feedback by mapping users and items to a latent factor space, so that user-item relationships (e.g., ratings) can be captured by their latent factors' dot product. Based on that, many research efforts have been made to enhance MF, such as integrating it with neighbor-based models[20] and extending it to factorization machines[33] for a generic modeling of features. Although they have shown good results, the recommendation performance of these methods will degrade significantly when the rating matrix is very sparse. Moreover, they can not provide explanations about why an item is worth recommending or not.

In the last few years, there is a large literature exploiting text review information for improving the rating prediction performance, such as HFT[27], RMR[26], EFM[44],TriRank[11], RBLT[39], and sCVR[32]. These work integrates topic models in their frameworks to generate the latent factors for users and items incorporating review texts. Specifically, EFM, TriRank and sCVR have been explicitly claimed that they can provide explanations for recommendations. These models first extract explicit product features (i.e. aspects) and user opinions by phrase-level sentiment analysis on user reviews, then generate feature-level explanations according to the specific product features to the user's interests. Besides, some studies focues on the preprocessing of reviews. The work in [45] is dedicated to filtering out the spam in reviews, and [17] utilizes supervised machine learning techniques to learn the "helpfulness" of reviews.

However, there are some limitations in these work. First, manual preprocessing is usually required for sentiment analysis and feature extraction of reviews[17, 32, 44]. Second, for EFM, TriRank and sCVR, the explanations are simple extractions of words or phrases from the texts, which changes the integrity of reviews and may distort their original meanings[34]. In contrast, we aim at making recommendations and selecting useful reviews simultaneously via an end to end neural network, which alleviates human effort

tremendously and provides more informative explanations. Another limitation of the above studies is that their textual similarity is solely based on lexical similarity[46]. The semantic meaning has been ignored in these works since the vocabulary in English is very diverse, and two reviews can be semantically similar even with low lexical overlapping. As a result, the approaches which employ topic modeling techniques suffer from a scalability problem.

Recently, deep neural networks have yielded an immense success in speech recognition, computer vision and natural language processing[9]. Some works also trying to combine different neural network structures with collaborative filtering to improve the recommendation performance. In[13], He et al. presented a Neural Collaborative Filtering (NCF) framework to learn the nonlinear interactions between users and items. Later, Neural Factorization Machines(NFM)[12] was developed to enhance FM by modelling higher-order and non-linear feature interactions. For review utilizing methods, Collaborative Deep Learning (CDL)[41] employs a hierarchical Bayesian model which jointly performs deep representation learning for the content and collaborative filtering for the rating matrix. DeepCoNN[46] uses convolutional neural networks to process reviews, and jointly models users and items by two parallel parts coupled by FM[33] in the last layer for rating prediction. NRT[25] combines gated recurrent neural networks and collaborative filtering to simultaneously predict ratings and generate abstractive tips simulating user experience and feelings. However, most of the existing methods failed to pay attention to the usefulness of reviews, which is the major focus of our work.

Attention mechanism has been shown effective in various machine learning tasks such as image/video captioning and machine translation[1, 4, 36, 42]. The key idea of soft attention is to learn to assign attentive weights (normalized by sum to 1) for a set of features: higher (lower) weights indicate that the corresponding features are informative (less informative) for the end task. In the field of recommendation systems, He et al. introduce an attention mechanism in CF which consists of both component-level and item-level attention module for multimedia recommendation[4]. [42] improves FM by discriminating the importance of different feature interactions via a neural attention network. In this paper, we use attention mechanism to learn the usefulness and contribution of each review to better model users and items for predicting item ratings and generating explanations.

3 PRELIMINARIES

3.1 Latent Factor Model

Before introducing our model, we begin by briefly introducing the Latent Factor Model[21]. LFM is a category of algorithms mostly based on matrix factorization techniques. One of the most popular algorithms of LFM predicts the rating $\widehat{R}_{u,i}$ of item i by user u as follows[39]:

$$\widehat{R}_{u,i} = q_u p_i^T + b_u + b_i + \mu \tag{1}$$

Here the equation contains four components: global average rating μ, user bias b_u, item bias b_i and interaction of the user and item $q_u p_i^T$. Further, q_u and p_i are K-dimensional factors that represent user preferences and item features, respectively.

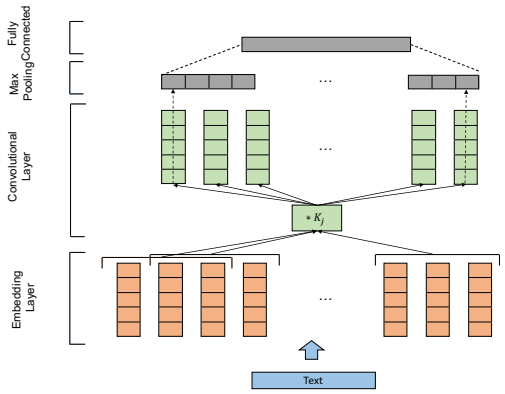

Figure 2: The CNN Text Processor architecture.

3.2 CNN Text Processor

In recent years, many text processing methods based on deep learning technology have been proposed and have achieved better performance than traditional methods. Such as fastText[15], TextCNN[18], TextRNN, and paragraph vector[23],etc. In this paper, we process text using the same approach as the current state-of-the-art method, DeepCoNN[46]. The method which is referred to as CNN Text Processor in the rest of this paper, follows TextCNN that inputs a sequence of words and outputs a n-dimensional vector representation for the input. Figure 2 gives the architecture of the CNN Text Processor.

In the first layer, a word embedding function $f : M \rightarrow \mathbb{R}^d$ maps each word in the review into a d dimensional vector, and then the given text will be transformed to a embedded matrix with fixed length T (padded with zero wherever necessary to tackle length variations). The embedding can be any pre-trained embedding like those trained on the GoogleNews corpus using word2vec[1][28], or on Wikipedia using GloVe[2][31].

Following the embedding layer is the convolutional layer. It consists of m neurons, and each associated with a filter $K \in \mathbb{R}^{t \times d}$ which produces features by applying convolution operator on word vectors. Let $V_{1:T}$ be the embedded matrix corresponding to the $T-$length input text. Then, j^{th} neuron produces its features as:

$$z_j = ReLU(V_{1:T} * K_j + b_j) \tag{2}$$

where b_j is the bias, $*$ is the convolution operation and $ReLU$[30] is a nonlinear activation function.

Let $z_1, z_2, ...z_j^{(T-t+1)}$ be the features produced by the j^{th} neuron on the sliding windows t over the embedded text. Then, the final feature corresponding to this neuron is computed using a max-pooling operation[7]. The idea behind max-pooling is to capture the most important feature-one with the highest value, which is defined as:

$$o_j = max(z_1, z_2, ...z_j^{(T-t+1)}) \tag{3}$$

The final output of the convolutional layer is the concatenation of the output from its m neurons, denoted by:

$$O = [o_1, o_2, ...o_m] \tag{4}$$

[1]https://code.google.com/archive/p/word2vec
[2]https://nlp.stanford.edu/projects/glove

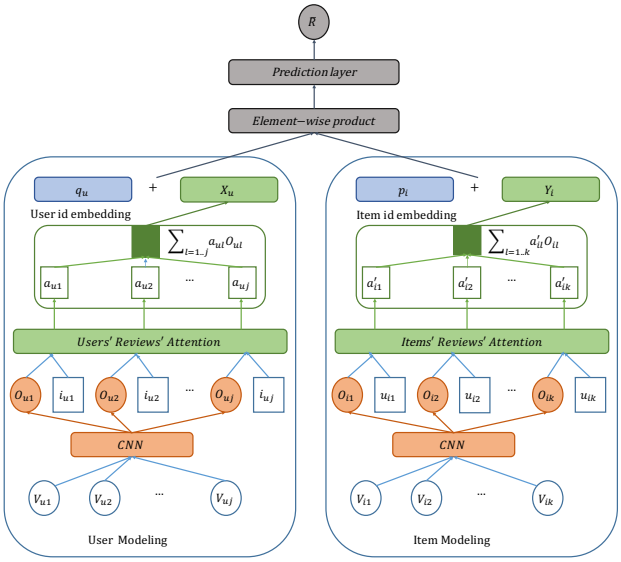

Figure 3: The neural network architecture of NARRE. Our attention model uses both IDs(i_{uj}, u_{jk}) and review content(O_{uj}, O_{jk}) to automatically assign weights to reviews.

Generally, the output O is then passed to a fully connected layer consisting of a weight matrix $W \in \mathbb{R}^{m \times n}$ and a bias $g \in \mathbb{R}^n$, which is:

$$X = WO + g \tag{5}$$

4 NEURAL ATTENTIONAL REGRESSION WITH REVIEWS

In this section, we introduce our Neural Attentional Regression with Reviews-level Explanation (NARRE). First, we will present the general architecture of NAMRE and the review processing method based on CNN Text Processor. Then, we will show our attention-based review pooling layer, which is the main concern in this paper. After that, we will introduce the prediction layer, which is a neural latent factor model designed for rating prediction. Lastly we will go through the optimization details of NARRE.

4.1 Overview of NARRE

The goal of our model is to predict a rating given a user and an item, as well as to select both useful and representative reviews. To this end, we utilize the attention mechanism to automatically assign weights to reviews when modeling users and items. The architecture of the proposed model is shown in Figure 3. The model consists of two parallel neural networks, one for user modeling (Net_u), and another for item modeling (Net_i). On the top of the two networks, a prediction layer is added to let the hidden latent factors of user and item interact with each other and calculate the final result of the model. At the training stage, the training data consists of users, items, and text reviews, while only users and items are available at the test stage. In the following, since Net_u and Net_i only differ in their inputs, we focus on illustrating the process for Net_i in details. The same process is applied for Net_u with similar layers.

In the first stage of Net_i, CNN Text Processor is applied to process the textual reviews of item i. We first discuss the limitation in the start-of-the-art model DeepCoNN[46], and then present our review processing method.

In DeepCoNN, all the reviews of item i are concatenated to a single matrix of world vectors V_i. In this case, item i's feature vector O_i is obtained directly by the convolutional layer (cf Eq. (2,3,4)). We argue that this method lead to loss of information. As the max-pooling operation is applied to the features generated from all the reviews of i, a strong feature in one review will override the rest of reviews[7]. But actually, it's not fair to judge an item from only one review in real life. In addition, max-pooling only keeps the maximum value, and even if a feature appears several times, it is only kept once. Hence the strength information of the feature is lost[16].

To alleviate the above problems, we propose to process each review of item i respectively. specifically, each review is first transformed into a matrix of word vectors, which we denoted as $V_{i1}, V_{i2}, ...V_{ik}$. Then, these matrixes are sent to the convolutional layer and the feature vectors of them can be obtained from the output. These feature vectors are noted as $O_{i1}, O_{i2}, ...O_{ij}$.

As these vectors are in the same feature space (they are all generated from the same convolutional layer), a general idea is to aggregate these vectors to get the representation of item i:

$$O_i = \sum_{l=1,\dots k} \frac{1}{k} O_{il} \tag{6}$$

However, Eq.(6) assumes that each review has the same contribution to item i, which is not robust in real life as the reviews are not equally useful and representative[45]. To settle this problem, we introduce attention mechanism into our model, which can help to learn the weight of each review in a distant supervised manner.

4.2 Attention-based Review Pooling

Attention mechanism has been widely used in many tasks, such as information retrieval[43], recommendation[4, 42], computer vision[5], and machine translation[8]. The goal of the attention-based review pooling in Net_i is to select reviews that are representative to item i's features and then aggregate the representation of informative reviews to characterize item i. A two-layer network is applied to compute the attention score a_{il}. The input contains the feature vector of the lth review of item i (O_{il}) and the user who wrote it (ID embedding, u_{il}). The ID embedding is added to model the quality of users, which helps identify users who always write less-useful reviews. Formally, the attention network is defined as:

$$a^*_{il} = h^T ReLU(W_O O_{il} + W_u u_{il} + b_1) + b_2 \tag{7}$$

where $W_O \in \mathbb{R}^{t \times k_1}$, $W_u \in \mathbb{R}^{t \times k_2}$, $b_1 \in \mathbb{R}^t$, $h \in \mathbb{R}^t$, $b_2 \in \mathbb{R}^1$ are model parameters, t denotes the hidden layer size of the attention network, and $ReLU$[30] is a nonlinear activation function.

The final weight of reviews are obtained by normalizing the above attention scores using the softmax function, which can be interpreted as the contribution of the lth review to the feature profile of item i:

$$a_{il} = \frac{exp(a^*_{il})}{\sum_{l=0}^{k} exp(a^*_{il})} \tag{8}$$

After we obtain the attention weight of each review, the feature vector of item i is calculated as the following weighted sum:

$$O_i = \sum_{l=1,\dots k} a_{il} O_{il} \tag{9}$$

The output of the attention-based pooling layer is a k_1 dimensional vector, which compresses all reviews of item i in the embedding space by distinguishing their contributions. Then it is sent to a fully connected layer with weight matrix $W_0 \in \mathbb{R}^{n \times k_1}$ and bias $b_0 \in \mathbb{R}^n$, which computes the final representation of item i:

$$Y_i = W_0 O_i + b_0 \tag{10}$$

4.3 Prediction Layer

In this paper, we apply our NARRE model for the recommendation task of rating prediction. To this end, we draw on the traditional latent factor model and extend it by the following ways: first, we extend user preferences and item features in LFM model to two components: one based on ratings while the other based on reviews. Then, inspired by[13], we present a neural form LFM for predicting ratings.

Specifically, the latent factors of user and item are first mapped to a shared hidden space. By introducing the latent representation learned from the reviews, the interaction between user u and item i is modelled as:

$$h_0 = (q_u + X_u) \odot (p_i + Y_i) \tag{11}$$

where q_u and p_i are user preferences and item features based on ratings as Eq. (1), X_u and Y_i are user preferences and item features obtained from the method introduced above, and $\odot$ denotes the element-wise product of vectors. Note that here the id embeddings are different from the id embeddings in Eq. (7). Because we think that user quality and preference are different kinds of objects with different characteristics. Modeling them in the same vector space would lead to limitations.

The output of Eq. (11) is a n dimensional vector, then it is passed to the prediction layer to get a real-valued rating $\widehat{R}_{u,i}$:

$$\widehat{R}_{u,i} = W_1^T h_0 + b_u + b_i + \mu \tag{12}$$

where $W_1 \in \mathbb{R}^n$ denotes edge weights of the prediction layer, b_u, b_i and μ denote user bias, item bias and global bias respectively. Clearly, by fixing W_1 to 1 and leave out X_u and Y_i, we can exactly recover the latent factor model. It is also obvious that we can add more hidden layers of non-linear transformation between h_0 and prediction layer, we leave the exploration as a future work.

4.4 Learning

Since the task we focus in this paper is rating prediction, which actually is a regression problem. For regression, a commonly used objective function is the squared loss[12, 25, 42]:

$$L_r = \sum_{u,i \in \mathcal{T}} (\widehat{R}_{u,i} - R_{u,i})^2 \tag{13}$$

where $\mathcal{T}$ denotes the set of instances for training, and $R_{u,i}$ is the ground truth rating assigned by the user u to the item i.

To optimize the objective function, we adopt the Adaptive Moment Estimation (Adam)[19] as the optimizer. Its main advantage is that the learning rate can be self adapted during the training phase, which eases the pain of choosing a proper learning rate and leads to faster convergence than the vanilla SGD.

Overfitting is a perpetual problem in optimizing a ML model. Many works have mentioned that deep learning models are even more likely to suffer from overfitting[9, 12, 18]. To alleviate this issue, we consider dropout[37] — a widely used method in deep learning models, in our work. The idea of dropout is randomly drop some neurons (along with their connections) during the training process[37]. When updating parameters, only part of them will be updated. Trough this process, it can prevent complex co-adaptations of neurons on training data. Moreover, as dropout is disabled during testing and the whole network is used for prediction, dropout has another side effect of performing model averaging with smaller neural networks, which may potentially improve the performance[42].

Specifically, in NARRE, we propose to adopt dropout on the attention based review pooling layer. After obtaining O_i which is a k_1-dimensional vector of latent factors, we randomly drop ρ percent of latent factors, where ρ is termed as the dropout ratio. Moreover, we also apply dropout after obtaining h_0 at the same way to prevent overfitting.

Table 1: Statistical details of the datasets.

	Toys_and_ Games	Kindle_Store	Movies_and_ TV	Yelp_2017
users	19,412	68,223	123,960	199,445
items	11,924	61,935	50,052	119,441
ratings & reviews	167,597	982,619	1,679,533	3.072.129

5 EXPERIMENTS ON RATING PREDICTION

5.1 Experimental Settings

5.1.1 ***Datasets.*** In our experiments, we use four publicly accessible datasets from different domains to evaluate our model. Three datasets are from Amazon 5-core[3][10]: **Toys_and_Games**, **Kindle_Store**, and **Movies_and_TV**. These datasets are selected to cover both different domains and different scales. Among them, Movies_and_TV is the largest dataset and it contains more than 1.6 million reviews, while Toys_and_Games is the smallest one and only contains about 160 thousand reviews.

Another dataset is from **Yelp** Challenge 2017[4]. It is a large-scale dataset consisting of restaurant ratings and reviews. Since the raw data is very large and sparse, we preprocessed it to ensure that all users and items have at least five ratings. Even so, it's still the largest dataset in all of our datasets. It contains more than 3 million reviews from about 200 thousand users.

The ratings of these datasets are integers in the range of $[1, 5]$. Since the length and the number of reviews have a long tail effect, we only keep the length and the number of reviews covering p percent users and items respectively, while p is set to 0.9 for Toys_and_Games and Kindle_Store, 0.85 for Movies_and_TV and Yelp. The characteristics of our datasets are summarized in Table 1.

5.1.2 ***Baselines.*** To evaluate the performance of rating prediction, we compare NARRE with five state-of-the-art models, namely

[3]http://jmcauley.ucsd.edu/data/amazon
[4]https://www.yelp.com/dataset_challenge

Table 2: Comparison of the Approaches

Characteristics	PMF	NMF	SVD++	HFT	DeepCoNN	NARRE
Ratings	√	√	√	√	√	√
Textual Reviews	\	\	\	√	√	√
Deep Learning	\	\	\	\	√	√
Review Usefulness	\	\	\	\	\	√

PMF, NMF, SVD++, HFT, and DeepCoNN. The first two methods only utilize ratings at the training stage, while the later two are representative review utilizing methods for rating prediction. The characteristics of the comparative approaches are listed in Table 2.

- **PMF**[29]: Probabilistic Matrix Factorization. Gaussian distribution is introduced to model the latent factors for users and items.
- **NMF**[24]: Non-negative Matrix Factorization.It only uses the rating matrix as the input.
- **SVD++**[20]: It extends Singular Value Decomposition with neighborhood models, in which a second set of item factors is added to model the item-item similarity.
- **HFT**[27]: This is the state-of-the-art method that combines reviews with ratings. An exponential transformation function is used to link the stochastic topic distribution in modeling the review text and the latent vector in modeling the ratings.
- **DeepCoNN**[46]: This is the state-of-the-art method that utilizes deep learning technology to jointly model user and item from textual reviews. The authors have shown that it can achieve significant improvements compared with other strong topic modeling based methods. In this paper, we implement this model and change the optimizer to Adam[19] as it can get a better performance than RMSprop[14] used in[46].

5.1.3 ***Evaluation Metric.*** To evaluate the performance of all algorithms, we calculate Root Mean Square Error (RMSE), which is widely used for rating prediction in recommender systems. A lower RMSE score indicates a better performance. Given a predicted rating $\widehat{R}_{u,i}$ and a ground-truth rating $R_{u,i}$ from the user u for the item i, the RMSE is calculated as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i}(\widehat{R}_{u,i} - R_{u,i})^2} \tag{14}$$

where N indicates the number of ratings between users and items.

5.1.4 ***Experiments Details.*** We randomly split the dataset into training (80%), validation (10%), and test (10%) sets. The validation set was used for tuning hyper-parameters and the final performance comparison was conducted on the test set. The parameters for the baseline algorithms were initialized as in the corresponding papers, and were then carefully tuned to achieve optimal performance. For deep learning based methods DeepCoNN and NARRE, the learning rate was searched in $[0.005, 0.01, 0.02, 0.05]$. To prevent overfitting, we turned the dropout ratio in $[0.1, 0.3, 0.5, 0.7, 0.9]$. The batch size was tested in $[50, 100, 150]$ and the latent factor number was tested in $[8, 16, 32, 64]$. After the turning process, we set the number of latent factors $k = 10$ for NMF and SVD++. We set the number of topics $K = 50$ for HFT. For CNN Text Processors in DeepCoNN and NARRE, we reuse most of the hyper-parameter settings reported by the authors of DeepCoNN since varying them did not give any perceivable improvement, the number of neurons, m, in the convolutional layer is 100, the window size t is 3. Moreover, we used a pre-trained word embeddings which are trained on more than 100 billion words from Google News[28]. Without special mention, we show the results of latent factor number n=32 and dropout ratio ρ=0.5 for DeepCoNN and NARRE.

5.2 Comparative Analysis on Overall Performances

The rating prediction results of our model NARRRE and baseline models on all datasets are given in Table 3. From the results, several observations can be made:

First, methods considering reviews (HFT, DeepCoNN and NARRE) generally perform better than collaborative filtering models (e.g. PMF, NMF and SVD++) which only consider the rating matrix as the input. This is not surprising, as review information is complementary to ratings and it can be used to improve the representation quality of latent factors. Therefore, the better-quality modeling increases the learning accuracy of user preferences and item features, and thus leads to a better rating prediction result.

Second, the methods that utilize deep learning technology (DeepCoNN and NARRE) usually outperform traditional methods, including HFT which also considers reviews for user and item modeling. We think that the reasons are as follows. First, previous works[18, 46] have shown that neural networks like CNN can be used to get better performance than topic models like LDA[2] in analyzing text information. Second, deep learning can model users and items in a non-liner way[13], which is the limitation of traditional CF-based models. What's more, some tricks in deep learning like dropout can be adopted to avoid overfitting and potentially improve the performance.

Third, as shown in Table 3, our method NARRE consistently outperforms all the baseline methods. Although review information is useful in recommendation, the performance can vary depending on how the review information is utilized. In our model, we propose a new attention based pooling for utilizing reviews while the representativeness of each review is considered. The representativeness allows each review to be modeled with a finer granularity, which can lead to a better performance according to the results.

5.3 Parameter Sensitivity Analysis

In this section, we show our exploration of parameters on the validation sets. Due to the space limitation, we only show the results of Toys_and_Games and Kindle_Store. The results of other datasets are similar to that of Kindle_Store. To better demonstrate the performance and improvement of our model, we extend DeepCoNN by changing its share layer from FM to our neural prediction layer (cf Eq. (11,12)), and name it DeepCoNN++. The results of DeepCoNN++ are also shown in the following figures.

We first explore the effect of the number of predictive factors. For MF methods (PMF, NMF and SVD++), the number of predictive factors is equal to the number of latent factors. Due to the weak performance of PMF and NMF, they are omitted in Figure 4 to better highlight the performance difference of other methods.

Generally, we can see that NARRE achieves the best performance on both datasets and all the predictive numbers. What's more, DeepCoNN++ outperforms DeepCoNN but still weaker than NARRE. This demonstrates the benefits of both the attention based

Table 3: Performance comparison on four datasets for all methods (RMSE). * and ** denote the statistical significance for $p < 0.05$ and $p < 0.01$, respectively, compared to the best baseline.

	Toys_and_Games	Kindle_Store	Movies_and_TV	Yelp-2017
PMF	1.3076	0.9914	1.2920	1.3340
NMF	1.0399	0.9023	1.1125	1.2916
SVD++	0.8860	0.7928	1.0447	1.1735
HFT	0.8925	0.7917	1.0291	1.1699
DeepCoNN	0.8890	0.7875	1.0128	1.1642
NARRE	**0.8769****	**0.7783****	**0.9965****	**1.1559***

Table 4: Examples of the high-weight and low-weight reviews selected by our model (a_{ij} means attention weight).

Item 1	a (a_{ij}=0.1932)	These brushes are great quality for children's art work. They seem to last well and the bristles stay in place very well even with tough use.
	b (a_{ij}=0.0161)	I bought it for my daughter as a gift.
Item 2	a (a_{ij}=0.2143)	From beginning to end this book is a joy to read. Full of mystery, mayhem, and a bit of magic for good measure. Perfect flow with excellent writing and editing.
	b (a_{ij}=0.0319)	I like reading in my spare time, and I think this book is very suitable for me.

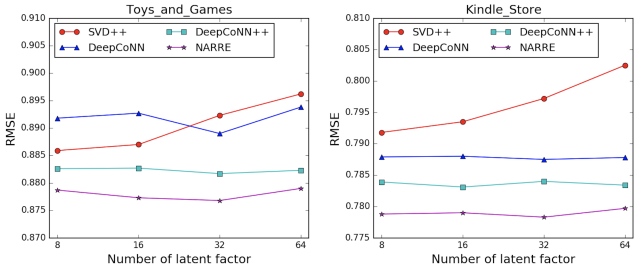

Figure 4: Performances w.r.t. different predictive factors (The number of dimensions of h_0 in Eq. (11)).

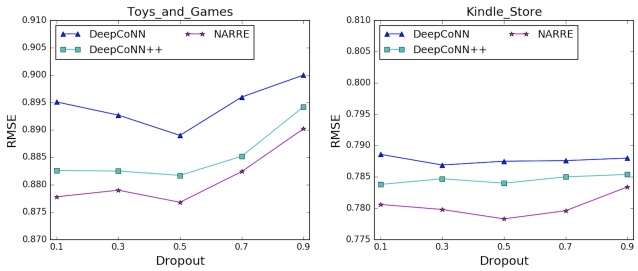

Figure 5: Performances w.r.t. different dropout ratios.

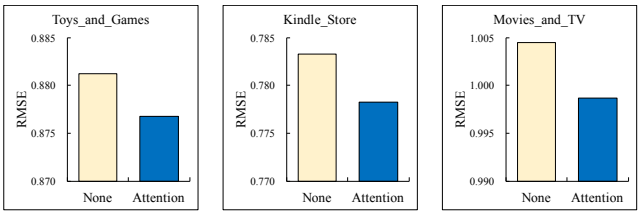

Figure 6: Effect of attention mechanism. The performances of the method with attention mechanism are significant better (p<0.05) than that of the method without it.

review pooling and our neural prediction layer. In addition, with the increase of the number of predictive factors, the performance of SVD++ decreases significantly (note that a higher RMSE value means a weaker performance), but for other methods it does not improve the performance or the opposite obviously.

We then study the effect of dropout on deep learning based methods. Figure 5 shows the validation performances of NARRE, DeepCoNN and DeepCoNN++ w.r.t. different dropout ratios.

From the results, we find that by setting the dropout ratio to a proper value, all methods can be significantly improved. This demonstrates the ability of dropout in preventing overfitting and as such, better generalization can be achieved. The optimal dropout ratio for NARRE is 0.5 on both datasets. Specifically, we find that the results of Toys_and_Games are more sensitive to the dropout ratios than the results of Kindle_Store. The reason we think is that the first dataset is very small, which makes the model more likely to overfit without dropout.

5.4 Effect of Attention Based Review Pooling

We now focus on analyzing the effect of attention based review pooling. Recall that to generate the attention weight of each review in Eq. (7), we incorporate different information sources. Specifically, in Net_i they are the review content and the id embedding of the user who wrote this review. In Net_u they are the review content and the id embedding of the item this review writing for. Note that when we do not consider attention, a normalized constant weight will be assigned to each review (Eq. (6)). The results are shown in Figure 6.

From the figure, we can see that when the attention mechanism is applied, the performance of rating prediction is improved significantly as compared with a constant weight method. It justifies our assumption that the usefulness of reviews are varied, and different reviews should have different representativeness for user preference and item feature. What' more, our attention-based review pooling can learn this representativeness well and lead to a better performance of the recommender algorithm.

5.5 Case Analysis

We provide some examples on the reviews and their final attention weights in Table 4 to illustrate the results on review usefulness identification. In the table, Review 1a and 2a represent more useful reviews with higher attention weights, and Review 1b and 2b are

Table 5: Usefulness evaluation on Amazon datasets (taking rated usefulness of reviews as ground truth). **: p<0.01 in statistical significance test, compared to the best baseline.

	Toys_and_Games				Kindle_Store				Movies_and_TV			
	Latest	**Random**	**Length**	**NARRE**	**Latest**	**Random**	**Length**	**NARRE**	**Latest**	**Random**	**Length**	**NARRE**
Precision@1	0.1487	0.3255	0.2476	**0.3860****	0.2447	0.4574	0.4041	**0.5235****	0.3040	0.4908	0.3903	**0.6576****
Recall@1	0.0362	0.0952	0.0771	**0.1398****	0.0400	0.0992	0.0852	**0.1131****	0.0436	0.0976	0.0677	**0.1445****
Precision@10	0.1550	0.2000	0.2316	**0.2697****	0.2228	0.2707	0.2933	**0.3530****	0.2325	0.2925	0.3369	**0.3459****
Recall@10	0.4367	0.5763	0.6763	**0.8601****	0.4510	0.5551	0.6168	**0.8317****	0.3716	0.4673	0.5403	**0.7674****

less helpful ones with lower attention weights. As we can see, the reviews with high attention weight generally contain more details about the item. For instance, from Review 1a and 2a, we can easily get the characteristics of each item, which is highly instructive for us to make our purchasing decisions. In contrast, the low attention reviews of the two items only contain the authors' general opinions, but show none or less details of the item. In fact, this kind of reviews is not very convincing to other users on making decisions.

6 EXPLAINABILITY STUDY

6.1 Review-level Explanation

Reviews are used to express users' experience and feelings, which also provide other users with detailed information and suggestions to help them make informed decisions. Since our NARRE model can simultaneously learn the weight of each review. The high-weight reviews contain more representative information of items, which is not only useful for item modelling, but also useful for users' reference. Therefore, by providing users with the highly-useful reviews, the explainability of recommender system is improved.

In fact, there are some e-commerce sites who have adopted this kind of explanation approach. However, in their settings, the reviews are selected by two simple methods. In the first method (named **Latest**), the reviews of an item are ranked by their writing time and the latest reviews are on the top. In the second method (named **Top_Rated_Useful**), each review is ranked by the number of users who have rated the review as useful. We argue that both the two methods have their deficiencies. The first method always puts the early reviews behind, which makes them hard to be found by users. The second method is not fair for new reviews since they haven't been rated. Moreover, the second method needs manual operation of users and the number of rated reviews usually very sparse in real life. Compared with them, NARRE can automatically select useful reviews when making recommendation, and doesn't suffer from the the above deficiencies.

In recent years, there are many works utilizing reviews for generating feature/word-level recommendation explanations. However, as natural language is susceptible to its integrity, the information of the review is not equal to the simple combination of features. Besides, the feature-level is not conflict with the review-level, which can also learned by attention mechanism[4]. We leave the exploration of feature-level explanations as a future work.

6.2 Usefulness in Terms of User Rated

As there are some reviews who have been rated useful by previous users, we first take these rated usefulness of reviews as ground truth to study the performance of NARRE in selecting useful reviews. Specifically, for each dataset, we only keep the items who have at least 1 rated review. For each item, the reviews are selected by four methods respectively, which are **Random** (The reviews are selected randomly), **Latest** (The most latest K reviews are selected), **Length** (The longest K reviews are selected) and **NARRE** (The K reviews with highest attention weight are selected). To evaluate the performance, we calculated the $Precision@K$ and $Recall@K$ as follows:

$$Precision@K = \frac{\sum_{j=1}^{K} rel_j}{K}; \quad Recall@K = \frac{\sum_{j=1}^{K} rel_j}{Re_i^{rated}} \tag{15}$$

where rel_j = 1/0 indicates whether the review at rank j in the Top-K list have been rated useful. To evaluate different length of review list, we set K=1 and 10. The results are shown in Table 5.

From the Table, we can see that when taking the rated usefulness of reviews as ground truth, the results of NARRE are significantly better than the other three methods. It shows that the attention weights are consistent with the users' perceptions of reviews. By applying attention mechanism, the usefulness of reviews can be learned well by the model.

Annotation Instructions 1:
Background: You are going to buy an item, so you want to refer to the reviews written by previous consumers to know more about this item.
Task1: You need to browse each of the reviews below and then determine whether it is useful for your purchasing.
The review can be classified as follows:
- **1 star**: Not useful at all.
- **2 stars**: Somewhat useful.
- **3 stars**: Fairly useful.
- **4 stars**: Very useful.

Figure 7: Review-level annotation instructions.

Table 6: Statistics of review-level annotation data.

	Items	Reviews	Reviews of each method	Annotations	Weighted κ
U_a	100	1264	745	3792	0.4112

6.3 Crowd-sourcing based Usefulness Evaluation

6.3.1 ***Review-level Usefulness Analysis.*** To study the effectiveness of our model's explanations in real life, we make crowdsourcing evaluation via CrowdFlower[5] platform to generate usefulness annotations for reviews. Specifically, for review-level annotations (U_a), we first randomly select 100 items for evaluation; for each item, we select top 10 reviews by each method, and generate the annotation pool by two methods, namely **Top_Rated_Useful** and **NARRE**. To make the experiment fair, for those items who have less than 10 Top_Rated_Useful reviews, we only use the same number of reviews by NARRE.

[5]https://www.crowdflower.com

Table 7: Performance comparison between NARRE and Top_Rated_Useful on annotation data, **: p<0.01.

	Precision@1	Precision@5	Precision@10	Recall@1	Recall@5	Recall@10	NDCG@1	NDCG@5	NDCG@10
Top_Rated_Useful	0.4800	0.4440	0.3610	0.0821	0.3453	0.4953	0.6640	0.6906	0.7076
NARRE	**0.5900****	**0.4760****	**0.3850****	**0.1067****	**0.3532****	**0.5046****	**0.7413****	**0.7231****	**0.7358****

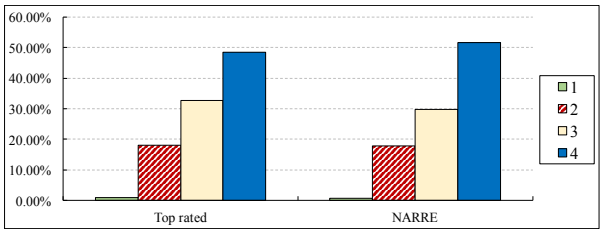

Figure 8: The distributions of the usefulness annotations (U_a) for the selected reviews. For usefulness, U_a = 1: not useful at all; 2: somewhat useful; 3: fairly useful; 4: very useful.

Figure 7 shows the task description and the instruction to assessors. To make sure the data annotations are reliable, we ensured that each task was judged by at least 3 different assessors. As the annotations of U_a are *ordinal*, we applied Cohen's Weighted κ[6] to assess the inter-assessor agreements. We chose to use the difference on the ordinal scale as the values in W. The statistics of annotation data are shown in Table 6. According to Landis et al. [22][6], moderate inter-assessor agreements between assessors are reached for U_a, which indicates the annotation data are of reasonable quality. The final "usefulness" of the reviews are calculated by averaging the three annotations and rounding them into integers.

Based on the data collected, we first investigated the usefulness distributions of the reviews selected by NARRE and Top_Rated_Useful respectively. In the following, we use the annotations (U_a) as the ground truth labels for usefulness. The distributions are shown in Figure 8. From the figure, we can see that the distributions of the two methods are similar, and almost 50% reviews are annotated as very useful (U_a=4). This is not surprising because all these reviews ranked in high positions by NARRE and Top_Rated_Useful respectively. Meanwhile, we notice that NARRE can select more "very useful" reviews compared to the latter (51.68% compared to 48.46%).

We then calculated the *Precision@K* and *Recall@K* of the "very useful" reviews (U_a=4) selected by the two methods. Besides, to further evaluate the ranking performance of each method, we also calculated the *NDCG@K* as follows:

$$DCG@K = \sum_{j=1}^{K} \frac{2^{rel_j} - 1}{log_2(j+1)}; \quad NDCG@K = \frac{DCG@K}{IDCG@K} \qquad (16)$$

while rel_j=[1–4] indicates the usefulness score of the review at rank j. The results are shown in Table 7. We can see that NARRE consistently and significantly better than the Top_Rated_Useful method in different evaluation metrics. Especially, great improvements are observed on the top one review performance. It indicates the effectiveness of our model in selecting high-quality and useful reviews.

[6]Landis et al. [22] characterize κ values < 0 as no agreement, 0–0.20 as slight, 0.21–0.40 as fair, 0.41–0.60 as moderate, 0.61–0.80 as substantial, and 0.81–1 as almost perfect agreement.

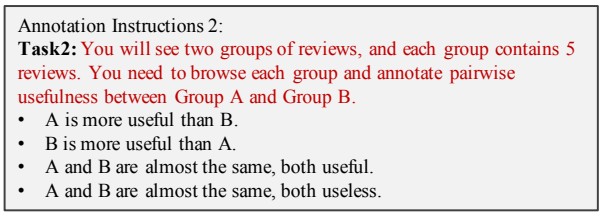

Figure 9: Pairwise annotation instructions.

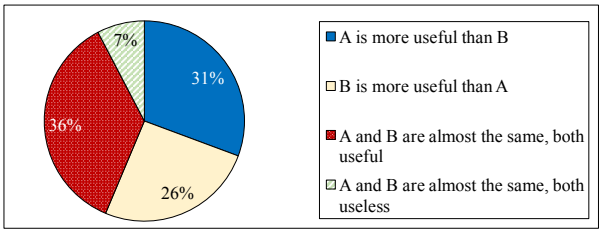

Figure 10: Pairwise annotations results (U_b). Group A: top 5 reviews by NARRE, Group B: top 5 reviews by Top_Rated_Useful.

6.3.2 ***Pairwise Usefulness Analysis.*** It is intuitive to conduct pairwise evaluation on whether our selected useful reviews are as helpful as the rated useful ones in the e-commerce system. Hence crowd-sourcing based pairwise experiments have been made (U_b). We randomly selected 100 items with two groups of reviews for comparison, namely reviews generated by **NARRE** and **Top_Rated_Useful** respectively. Since the average number of useful reviews for each item by the system user ratings in the dataset is 3.71, and 21.50% items with usefulness ratings have more than 5 useful reviews, to make a fair comparison, each group only contain top 5 reviews for each item. (The items with no less than 5 rated useful reviews have been used.) The order of the two review groups are randomly shown in the experiments. The instructions are shown in Figure 9. Each pair of reviews is judged by 3 assessors. The Kappa score for consistency is 0.2981, showing fair agreements for U_b.

Pairwise evaluation results are shown in Figure 10. For 31% of items, our selected reviews are thought of more useful than the top usefulness-rated ones. While only for 26% of items, NARRE is thought weaker. In addition, for 36% of items, the annotators find that both groups of reviews are useful and it is hard to judge which is better. Therefore we can conclude that in most cases, NARRE achieves equal or even better performance than system's usefulness rating method in selecting useful reviews. Note that in this experiment, we only select the items who have at least 5 usefulness-rated reviews. according our observation, only 37.51% items have user ratings on review usefulness in the dataset. Since this Amazon dataset has been filtered before its release to public, the ratio of usefulness-rated reviews is even sparser in real applications. For these majority part of items that lack of review usefulness rating information, our proposed NARRE approach will be great helpful on

selecting useful reviews to the users for their purchasing decision making in real e-commerce scenarios.

7 CONCLUSION

Post-purchase reviews play a very import role for user's purchasing behavior. However, it is hard for users to find useful information from an immense amount of reviews. In this paper, we propose a neural attentional model named NARRE which simultaneously predicts precise user ratings to the item and select useful reviews automatically to provide review-level explanations. Extensive experiments have been made on 4 real-life datasets from Amazon and Yelp. In terms of recommendation performance, the proposed NARRE consistently outperforms the state-of-the-art recommendation models based on matrix factorization and deep learning in rating prediction. In terms of review usefulness identification, the highly-useful reviews selected by NARRE are consistent with, if not better than, users' usefulness rating in e-commerce system. Furthermore, it contributes a lot to the system when there is no review-usefulness-labeling available, which is common in real e-commerces. Therefore, our proposed model NARRE will help build a more efficient and explainable recommender system.

This review-level usefulness identification can also help other researches on feature/word-level explainable recommendation, e.g. models in [11, 32, 44], as the pre-selection of reviews, which will be studied in the future. Feature-level attention model is also the potential future work. Moreover, we are interested in exploring whether our method can used to model user quality and help identify users who always write less-useful reviews.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the Natural Science Foundation of China under Grant No.: 61672311 and 61532011.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[3] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. *arXiv preprint arXiv:1704.02298* (2017).
[4] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*. 335–344.
[5] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, and Tat-Seng Chua. 2016. SCA-CNN: Spatial and Channel-wise Attention in Convolutional Networks for Image Captioning. *arXiv preprint arXiv:1611.05594* (2016).
[6] J Cohen. 1968. Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin* 70, 4 (1968), 213.
[7] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.
[8] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *SIGKDD*. 193–202.
[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.
[10] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*. 507–517.
[11] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*. 1661–1670.
[12] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. (2017).
[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*. 173–182.
[14] G Hinton, N Srivastava, and K Swersky. 2012. RMSProp: Divide the gradient by a running average of its recent magnitude. *Neural networks for machine learning, Coursera lecture 6e* (2012).
[15] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* (2016).
[16] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. *Eprint Arxiv* 1 (2014).
[17] Soo Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. 2006. Automatically assessing review helpfulness. In *EMNLP*. 423–430.
[18] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).
[19] D Kinga and J Ba Adam. 2015. A method for stochastic optimization. In *ICLR*.
[20] Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *SIGKDD*. 426–434.
[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[22] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.
[23] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. 4 (2014), II–1188.
[24] Daniel D Lee and H Sebastian Seung. 2001. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*. 556–562.
[25] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural Rating Regression with Abstractive Tips Generation for Recommendation. (2017).
[26] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*. 105–112.
[27] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.
[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. 3111–3119.
[29] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.
[30] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*. 807–814.
[31] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *EMNLP*. 1532–1543.
[32] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social collaborative viewpoint regression with explainable recommendations. In *WSDM*. 485–494.
[33] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.
[34] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *SIGKDD*. 1135–1144.
[35] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[36] Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685* (2015).
[37] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research* 15, 1 (2014), 1929–1958.
[38] Xiaoyuan Su and Taghi M Khoshgoftaar. 2009. A survey of collaborative filtering techniques. *Advances in artificial intelligence* 2009 (2009), 4.
[39] Yunzhi Tan, Min Zhang, Yiqun Liu, and Shaoping Ma. 2016. Rating-Boosted Latent Topics: Understanding Users and Items with Ratings and Reviews.. In *IJCAI*. 2640–2646.
[40] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *IUI*. 47–56.
[41] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*. 1235–1244.
[42] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *arXiv preprint arXiv:1708.04617* (2017).
[43] Chenyan Xiong, Jimie Callan, and Tie-Yen Liu. 2017. Learning to attend and to rank with word-entity duets. In *SIGIR*.
[44] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.
[45] Yongfeng Zhang, Yunzhi Tan, Min Zhang, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2015. Catch the Black Sheep: Unified Framework for Shilling Attack Detection Based on Fraudulent Action Propagation.. In *IJCAI*. 2408–2414.
[46] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*. 425–434.