Distributed Representations for Building Profiles of Users and Items from Text Reviews

Wenliang Chen[†], **Zhenjie Zhang**[‡], **Zhenghua Li**[†], **Min Zhang**[†]
[†]School of Computer Science and Technology, Soochow University, China
[‡]Advanced Digital Sciences Center, Illinois at Singapore Pte. Ltd., Singapore
{wlchen, zhli13, minzhang}@suda.edu.cn
zhenjie@adsc.com.sg

Abstract

In this paper, we propose an approach to learn distributed representations of users and items from text comments for recommendation systems. Traditional recommendation algorithms, e.g. collaborative filtering and matrix completion, are not designed to exploit the key information hidden in the text comments, while existing opinion mining methods do not provide direct support to recommendation systems with useful features on users and items. Our approach attempts to construct vectors to represent profiles of users and items under a unified framework to maximize word appearance likelihood. Then, the vector representations are used for a recommendation task in which we predict scores on unobserved user-item pairs without given texts. The recommendation-aware distributed representation approach is fully supported by effective and efficient learning algorithms over massive text archive. Our empirical evaluations on real datasets show that our system outperforms the state-of-the-art baseline systems.

1 Introduction

With the prosperity of Internet-based e-commerce in the last decade, millions of item (e.g., product/serivce) comments are now available online and even more are flooding in an explosive manner. Yelp (www.yelp.com), as the largest restaurant comment web site, for example, attracts more than 140 million users to contribute new text comments every month, covering almost all restaurants in USA. It is widely believed that text comments contain much richer information on users as well as items than dull scores, with far more precise descriptions on personal preferences and item features. To fully unleash the potential value underneath the text comments, huge research efforts are now devoted to the design and development of new technologies to capture and understand the semantics in the comments, particularly those associated with users and items. The text analytical outcomes are expected to support decision making and provide new business opportunities in e-commerce.

Recommendation is recognized as one of the most important components in e-commerce systems, driving the growth of profits. Traditional recommendation systems rely on scores over user-item pairs under a bipartite graph model, such that accurate score prediction over unobserved user-item pair helps the system to identify potential interested buyers. A huge bulk of prediction and recommendation strategies are proposed in this domain, e.g. collaborative filtering and matrix completion (Sarwar et al., 2001; Koren et al., 2009). Because of the limited information included in each individual score, such recommendation strategies commonly suffer from the cold-start problem (Zhou et al., 2011), which returns poor recommendations to newcoming users. While text mining on comment archive may provide important information to the recommendation engine, even for fresh users, the integration of such information adds new complexity and challenges to the system, since the text mining results are not directly useful to the recommendation algorithms. In particular, most of the opinion mining techniques available in the literature (Liu and Zhang, 2012; Melville et al., 2002) do not distinguish attributes on user preference from attributes of items. The recommendation engines are thus unable to understand the underlying reasons behind the good and poor opinions from the users on the items. While there are

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2143–2153, Osaka, Japan, December 11-17 2016.

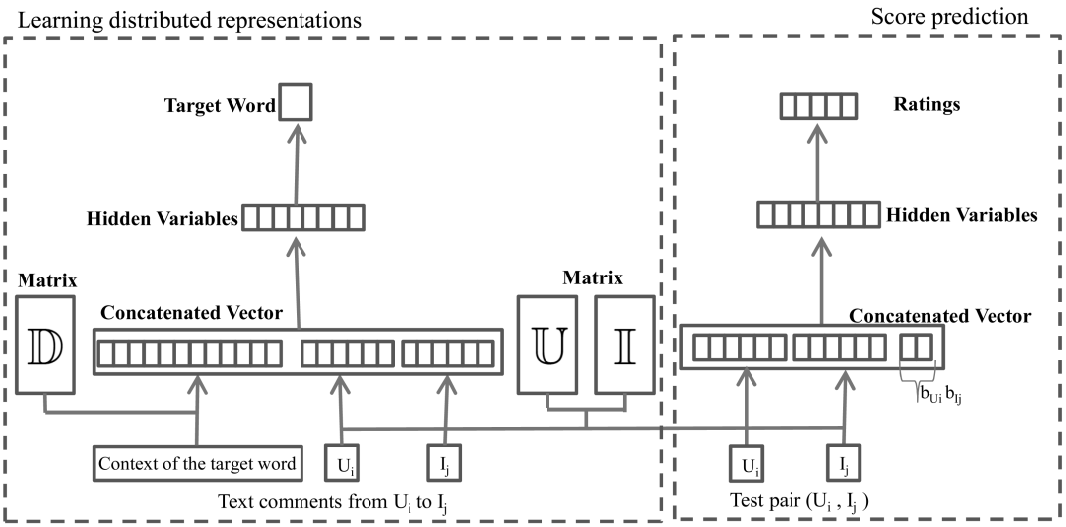

Figure 1: Our framework for learning distributed representations for recommendation

attempts on tightly connecting recommendation and text mining, e.g. (McAuley and Leskovec, 2013; Almahairi et al., 2015), these studies make strong assumptions on the generative mechanism behind the text comments and scores, which may not correctly reflect the interaction between user preferences and item attributes appropriately.

To better exploit the value of text comments for recommendation engine, we propose a simple yet effective approach inspired by the successful distributed representation models on text semantics extraction (Le and Mikolov, 2014; Mikolov et al., 2013). In our approach, we learn distributed representations from text comments to represent users and items. Our approach has a number of unique advantages, which make it a promising alternative to existing feature extraction methods used in comment-based recommendation systems. Firstly, the distributed representations on users and items are aligned, enabling the employment of a huge cluster of classification approaches on prediction and recommendation. Secondly, distributed representations do not rely on any assumption on the generative procedures over the user-generated contents. This feature generally avoids the potential bias on the analytical outcomes, caused by inappropriate setting on the priors, and finally saving computation overheads. Thirdly, the distributed representations work well on both long and short texts. It is thus likely to enhance the usefulness of the mining outcomes, by training over text data from multiple sources, e.g. food blogs and restaurant comments.

In our approach, we adapt the vectorization techniques used in the training of distributed representations over words and documents, to address the problem of factor decomposition between user preferences and item attributes. User vectors and item vectors are introduced as independent representations to the target users and items correspondingly and are learned by a neural network. In the model, the user vector is shared across the comments written by the same user and the item vector is shared across the comments written on the same item. Then, the vector representations are included in the training of the task of score prediction in which we predict rates on unobserved user-item pairs without given texts. The experimental results on real datasets show that our approach generates a significant margin of performance advantage over existing methods in the task of score prediction.

2 Preliminaries

In this section, we define the task of score prediction and introduce the background of distributed representations of words and documents.

2.1 Score Prediction

Assume that we have a fixed group of n users $\mathbb{U} = \{U_1, U_2, \ldots, U_n\}$ and a fixed group of m items $\mathbb{I} = \{I_1, I_2, \ldots, I_m\}$. Here, an item I_j can be either a product or a service, provided to all the users in $\mathbb{U}$ without any restriction. Each comment C_k in database $\mathbb{C} = \{C_1, C_2, \ldots, C_l\}$ composes a sequence of L_k words $\{w_1, w_2, \ldots, w_{L_k}\}$, with every word drawn from a known dictionary $\mathbb{D}$. Each comment C_k is also uniquely associated with a user $U_i \in \mathbb{U}$, an item $I_j \in \mathbb{I}$ and a score S_k. It indicates that user U_i purchases item I_j with score S_k and detailed text comments in C_k. The recommendation system is expected to return a group of items to each user U_i, such that the user U_i is more likely to buy these items with high scores and good comments. Therefore, the problem of recommendation is usually transformed into the score prediction problem, as formally defined below.

Problem 1 ***Score Prediction*** *Given $(\mathbb{U}, \mathbb{I}, \mathbb{C})$, the problem of score prediction is to estimate the score over unseen user-item pair (U_i, I_j) without text comments, where $U_i \in \mathbb{U}$ and $I_j \in \mathbb{I}$.*

Obviously, the score prediction is quite different from text-based sentiment classification which has text information in the testing stage. The recommendation system is more efficient, if the score prediction returns more accurate estimations.

2.2 Distributed Representations

Here, we briefly review the approaches of learning distributed representations of words and documents(Mikolov et al., 2013; Le and Mikolov, 2014). Word distributed representation denotes semantical meanings of the words extracted from large-scale text archive in a unified way (Mikolov et al., 2013). Each word is represented by a vector of fixed length L, such that 1) semantically similar words are close to each other in the new vector space; and 2) the semantics of words are compositional by vector operations. To simplify the notation, we use w to denote a word in the dictionary, as well as its corresponding vector representation. Given a sequence of words, i.e. $w_1, \ldots, w_T$, the distributed representations of the words are calculated by maximizing the likelihood of observing the words in the sequence, based on the neighbor words in the sequence. Mathematically, with the specified neighborhood width d, this objective is formalized as

$$\frac{1}{T}\sum_{i=1}^{T} \log \Pr\left(w_i \mid w_{i-d}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+d}\right).$$

For words at the beginning and end of the sequence, special null words are inserted as the padding words. To generate the vector representations of the words, the algorithm runs an optimization over the vectors as well as a group of weights, with a weight vector v_{w_i} of length $(2d-1)L$ associated with every word w_i. Given the representations of the words in form of vectors, the vector representations of the words in the neighborhood of w_i are concatenated, resulting in a long vector x_i of length $(2d-1)L$. The probability of observing w_i in the context of x_i is estimated by the following equation:

$$\Pr\left(w_i \mid x_i\right) = softmax(W \cdot x_i + b). \tag{1}$$

where W is the weight matrix connecting the input and the output and b is a bias vector. To efficiently optimize the weights and word representations, a number of optimization tricks, including hierarchical softmax and negative sampling are suggested in (Mikolov et al., 2013), in order to train the representations in reasonable time. The detail of learning procedure can be found in the paper of (Mikolov et al., 2013).

Paragraph (or document) vector is an extension of word distributed representation, by introducing a global vector indicating the topics of the whole paragraph (Le and Mikolov, 2014). When generating the representation of the neighborhood x_i, the paragraph vector, another vector of fixed length across paragraphs, is included. The paragraph vector is effective on capturing the context of the words, which can be used to enhance the accuracy of the word prediction model and extract overall topics of the whole paragraph.

3 Our Approach

Our approach for learning distributed representations is inspired by the models for learning the vectors of words and documents in (Le and Mikolov, 2014; Mikolov et al., 2013). In our solution, as presented in Figure 1, there are three types of distributed representations built and utilized in the learning phase, including word representation, user representation and item representation. Each representation is a vector of fixed length in its respective domain, say W, U and I respectively. Intuitively, the word representation denotes the conceptual meanings of the words in a latent space. Similarly, the user representation denotes the personal preferences over the items, and item representation denotes the important attributes of the items. The dimensions of the user representations, for example, are expected to indicate personal preference and behavior patterns, such as 1) whether the user prefers cheap items; or 2) whether the user prefers better service to good foods during dining in restaurants. We hereby emphasize that such preferences and profiles are automatically extracted from the data, without any supervision or manual labeling involved.

When the context is clear, we use U_i (resp. I_j) to denote both the user identity (resp. item identity) as well as its corresponding representation vector. Similarly, we misuse w to denote a word in dictionary $\mathbb{D}$ and its word representation vector. There are two parts in our approach: 1) Learning distributed representations of words, users, and items; 2) Score prediction for pairs of users and items without text comments.

3.1 The Learning Model

During the learning phase, the contexts are fixed-length and sampled from a sliding window over the text comments. The matrix $\mathbb{D}$, with stacked word vectors, is shared across all the text comments. However, the user vector is shared across the comments written by the same user and the item vector is shared across the comments written on the same item.

By looking up the matrices, we obtain the vectors of users, items, and words. After concatenating all relevant vectors in order, including user vector, item vector and neighborhood word vectors, a bi-layer neural network model is built on top of the concatenated vector to predict the words as shown in the left part of Figure 1.

Suppose that the target word is w and the input (the concatenated vector) is X_w. A intermediate layer is applied on the input vector X_w to generate a fixed length vector with L_h binary variables V_h. Each variable $v_i \in V_h$ is independently activated, based on the weight assignment on the edges between input vector and intermediate layer, following the logistic function:

$$\Pr(v_i = 1) = \frac{1}{1 + \exp\left(-\left(\sum_{x_j \in X_w} x_j \cdot w_{ij}\right)\right)}.$$

where w_{ij} is the weight connecting from the j^{th} dimension of the input to the i^th dimension of V_h. From the intermediate layer, our model predicts the word occurrence with another *prediction layer*. On this layer, the system employs exactly the same softmax function used in the existing word vector and paragraph vector models. The prediction function is,

$$\Pr\left(w \mid V_h\right) = softmax(W_w^h \cdot V_h + b^h).$$

where W_w^h is the weight vector connecting V_h to w and b^h is the bias vector.

Based on the model above, the training of the model with the text comment archive is in two folds. Firstly, the weights in the neural network are optimized to reflect the correlation between user/item/word representations and the word occurrence likelihood. Secondly, the construction of the representation identifies meaningful semantics into the vectors, especially for the user representation and item representation. When there are multiple users and items available in the comment database, the result representations automatically reveal the actual preferences of the users and the attributes of the items, because common topics across comments are merged and appear as a dimension in the representations.

3.2 Training the Parameters

During training, the parameters we should learn include the weights in the above equations and the vector representations of users, items and words. We use stochastic gradient descent to learn the parameters. Then the *backpropagation* algorithm is employed to push the gradients to the concatenated vector level, and thus used to update the weights on the edges as well as the user/item/word vectors by exactly the same mechanism. At every step of the learning phase, we sample a fixed-length context from a random comment.

To train the weights from hidden variables to the target words, we apply *negative sampling*. The negative sampling method is a simplified variation of Noise Contrastive Estimation (Gutmann and Hyvarinen, 2012; Mhih and Teh, 2012). To compute the probabilities efficiently, we use the negative sampling method proposed by (Mikolov et al., 2013), which approximates the probability by the correct example and K negative samples for each instance. The formulation to compute $\log(\Pr(w_i|V_h))$ is,

$$\log \sigma(z_{w_i}) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P(w)}[\log \sigma(-z_{w_k})] \tag{2}$$

where $\sigma(z_w) = 1/(1+exp(-z_w))$, $z_w = W_w^h \cdot V_h + b^h$, w_i is the target word, V_h is the vector of hidden variables, and $P(w)$ is the noise distribution on the data. We set K as 5 in our experiments as used in (Mikolov et al., 2013). We perform the iterative update after predicting the target word,

$$\theta \leftarrow \theta - \alpha(\frac{\partial \sum log(\Pr(w_i|V_h))}{\partial \theta}) \tag{3}$$

where α is the learning rate and θ is the set of parameters to learn that includes the weights of the model. The initial value of α is 0.025. During training, the learning rate is halved if the log-likelihood does not improve significantly after one update. Then, the training stops if it does not improve again.

The computation efficiency on the training procedure is excellent, because most of the computation time is spent on the training between hidden variables and target words. The training between concatenated vectors and hidden variables is much more efficient, since the number of variables involved is highly constrained, depending on the specified size of the word neighborhood and the size of the hidden variable layer.

After the learning phase, we obtain the distributed representations of users and items shown as vectors. They are used in the score prediction task.

3.3 Predicting Scores

As is described in Problem 1, the problem of score prediction is to estimate the score of a user before he/she actually purchases the item. Thus, we do not have the text comment for the given pair when testing. We build a regression/classification model on top of the user/item representations. Specifically, a prediction model is a parameterized function $F(U_i, I_j)$ with inputs of user/item vectors and outputs of score estimation. While any regression model can be employed as the parameterized function $F(\cdot)$, we use the network shown in the right part of Figure 1. First, we learn a non-linear transformation which can project the input user/item representations into a space where it becomes linearly separable. The space is a intermediate layer referred to as a hidden layer. Then, a logistic regression classifier takes the transformed vectors for predicting the scores.

As indicated in Figure 1, the input layer contain the user and item representations and the average scores (b_{U_i} and b_{I_j}) of user and item, and the output layer has possible ratings.

Given the input x, $h(x) = s(W^1 x + b^1)$ represents the hidden layer, where W^1 is the weight matrix connecting the input vector to the hidden layer and b^1 is a bias vector for the transformation. Here, we use $tanh(a) = (e^a - e^{-a})/(e^a + e^{-a})$ for function s since it is fast when training the parameters.

Specifically, the prediction is then obtained by applying:

$$P(y|x) = softmax(W^2 h(x) + b^2) \tag{4}$$

	Yelp	Dianping	Amazon
review#	1.6M	1.2M	1.2M
item#	60,785	43,141	246,200
users#	366,715	350,936	641,380
review# per item	25.81	27.98	4.73
review# per user	4.27	3.44	1.81
word# per review	128.93	91.78	167.39

Table 1: Statistics of data set

where W^2 is the weight matrix connecting the hidden layer to the output layer and b^2 is a bias vector for the classifier. We use stochastic gradient descent to learn all parameters of the model, including W^1, b^1, W^2 and b^2. Finally, the prediction of the model y_{pred} is the class whose probability is maximal:

$$y_{pred} = \text{argmax}_i P(Y = y_i|x) \tag{5}$$

4 Experiments

4.1 Dataset

We evaluate the systems on datasets from a variety of public sources. The first one is from Yelp Dataset Challenge[1], which contains about 1.6M reviews. The second one is from *Dianping* (www.dianping.com), which consists of user reviews on the restaurants located in China. We also include about 1.2M reviews from Amazon previously used for opinion analysis in (Jindal and Liu, 2008). In the datasets, each review contains a user ID, restaurant/product ID, numeric rating (from 1 to 5), and detailed comment text. We filter out the reviews that do not have comment texts. The detailed statistics of the data sets are summarized in Table 1. The reviews are written in two different languages, Yelp and Amazon using English, and Dianping using Chinese. The experiments with different languages and domains demonstrate the genericity of our approach.

For each dataset, we randomly split the whole set into two parts: 80% as training data and 20% as test data. We also randomly select 10% of training data as development data to tune the parameters of the systems.

4.2 Score Prediction

We report the mean absolute error (MAE) of the score predictor, i.e., $MAE = \frac{1}{N}\sum_{u,i\in T}|\hat{r}_{ui} - r_{ui}|$, where T is the test set, N is the total number of predicted ratings, $\hat{r}_{ui}$ and r_{ui} are the predicted rating and the user assigned rating scores for user u and item i, respectively.

We compare with four systems: 1) CF: This is a user-based Collaborative Filtering (CF) system in which we follow the description of (Sarwar et al., 2001). 2) FM: This is a latent factor model implemented in libfm[2](Rendle, 2012). 3) CTR: This is a collaborative topic regression model adapted from (Wang and Blei, 2011)[3], which is proposed to recommend scientific articles in citation networks. 4) HFT: This is the Hidden Factors as Topics (HFT) model proposed by (McAuley and Leskovec, 2013), which is a state-of-the-art system using text reviews[4]. The first two systems use rating information only and the other two systems utilize text reviews. The settings of all the systems are tuned on the development sets. According to the results on the development sets, we set user/item vectors size as 100 and number of hidden units as 50 in our system.

Table 2 shows the experimental results of score prediction, where DRT is our system. FM is better than the traditional Collaborative Filtering (CF) system. This shows that the factor model is very powerful. HFT and CTR achieve similar scores to FM on two datasets and perform better than FM on the Amazon

[1]http://www.yelp.com/dataset_challenge/

[2]http://libfm.org/

[3]https://www.cs.princeton.edu/∼chongw/software/ctr.tar.gz

[4]http://cseweb.ucsd.edu/∼jmcauley/code/code_RecSys13.tar.gz

data. Our system provides the best performance on all the datasets and is significantly better than other systems ($p < 10^{-5}$).

System	Yelp	Dianping	Amazon
CF	1.0004	0.6906	0.9924
FM	0.9276	0.6609	0.8936
CTR	0.9243	0.6619	0.8693
HFT	0.9221	0.6615	0.8658
DRT	0.9064	0.6316	0.8165

Table 2: Results of score prediction (MAE)

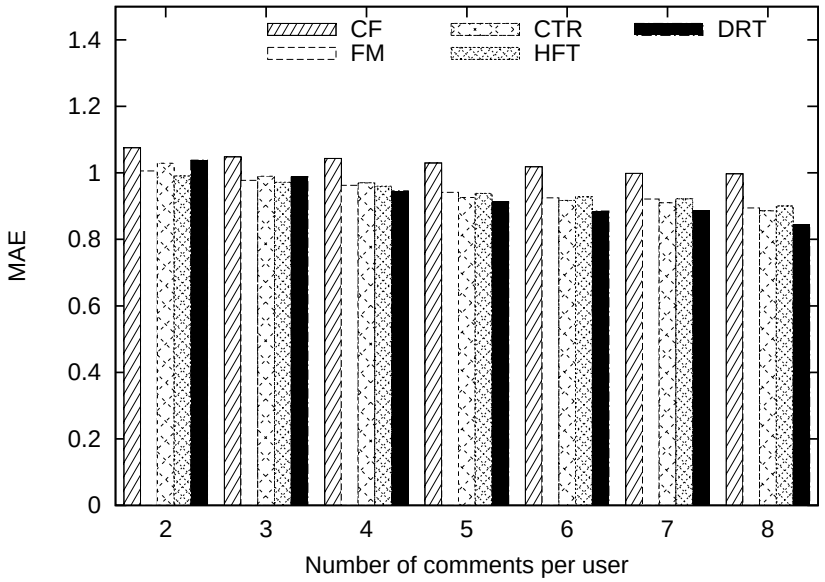

Figure 2: MAE vs number of training reviews per user (Yelp)

The number of training reviews is important for recommendation systems. We thus investigate the effect of training reviews per user. Figure 2- 4 show the MAE scores with varying number of training reviews per user. The trend of the results shows that all the systems are getting better scores, with increasing number of comments. When the number of training reviews grows larger, the performance gap between DRT and CF also grows. This fact indicates that our approach can make use of text comments to improve performance. The results also show that with 2 or 3 training reviews, DRT performs best on 3 among 6 cases, while with 4 or more training reviews, DRT can beat other approaches. In future work, we will try to improve the bad cases.

We also investigate the effect of different lengths of comments to our system (DRT). To reduce the effect of numbers of training reviews per user, we calculate the average lengths of comments written by the users who only have 4-10 training reviews. We ignore the users whose average comment length is no more than 20 words. We group the users into several groups by setting every 20 words as one BIN. For example, BIN-40 includes the users whose average comment length is in (20,40] and BIN-100 includes the users whose average comment length is in (80,100]. Figure 5 shows the relation between the MAE scores and the comment lengths. From the figure, we find that DRT roughly gets consistent performance along with the increase of comment lengths. However, this fact is against what we expect: the longer texts might result in better performance. We check some long comments in three datasets and find that most of the long comments indeed have more detailed information, but also bring some noise. For example, some users like to describe the experiences in other restaurants before commenting on the target restaurant. The results indicate that our system can partial reduce the effect of noisy information and achieve similar performance on both long and short texts.

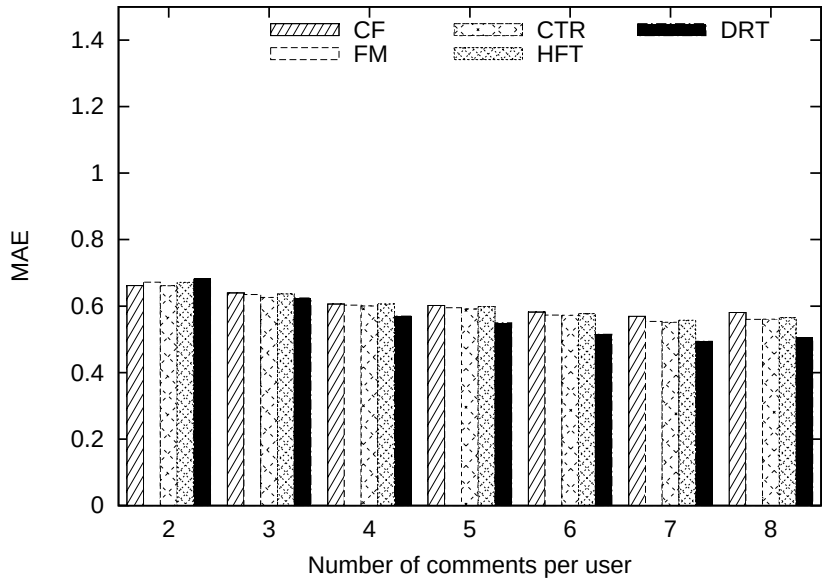

Figure 3: MAE vs number of training reviews per user (Dianping)

5 Related Work

In this paper, we utilize user comments to enhance recommendation systems. Text mining techniques are now widely used on comment data on the Internet to support a large variety of applications in e-commerce. Liu et al. (2005) discuss the possibility of comparing comments available online to identify important opinions of the users on the items. Zhai et al. (2011) present an approach to select important sentences in the product comments. Zhang et al. (2015) design a customized solution to restaurant comment summarization on dishes. Topic modeling techniques based on Dirichlet allocation are recently popular in text analytics (Blei et al., 2003), which is particularly effective on long documents to find meaningful topics (Blei et al., 2004). However, we hereby emphasize that all these techniques are not directly helpful to improve recommendation systems.

A huge bulk of recommendation algorithms are proposed in the literature. Collaborative filtering is known as the most popular approach (Su and Khoshgoftaar, 2009; Sarwar et al., 2001) and matrix factorization has recently emerged as an effective strategy to improve the effectiveness of collaborative filtering (Koren et al., 2009; Rendle, 2012). Latent Factorization Model decomposes the matrix of user-item features and identifies the features connected to users and items respectively. Some other invariants, such as max-margin matrix factorization (Rennie and Srebro, 2005) and probabilistic matrix factorization (Salakhutdinov and Mnih, 2008), are proposed to improve the robustness of the factorization outputs. Singh and Gordon (2008) further encode genres of movies and roles of actors in movies as binary relations into a collective matrix factorization model. All these approaches require the generation of matrix with aligned features, and only can use very limited information besides scores.

There are a handful of attempts to incorporate rich text information into recommendation engines. Musat et al. (2013) revise the original collaborative filtering approach, by modeling user similarity based on the feature words shown in their text comments. In their approach, the feature words are nouns with the highest opinion counts. Zhang et al. (2014) adopt a similar strategy to support personalized recommendation based on text comments. Their approach utilizes existing sentiment analysis tools (Lu et al., 2011) to generate the feature words. These words are fed into latent factor model (Koren et al., 2009) to build profiles on the users and items. However, the feature extraction procedure used in their method does not distinguish user preference and item features in the opinion. He et al. (2015) tries to build tripartite graph model with user, item and discussion topic engaged. The recommendation system exploits the tripartite graph model to make personalized recommendation decisions. In their approach, the feature words are identified similarly as Zhang et al. (2014). All the above studies

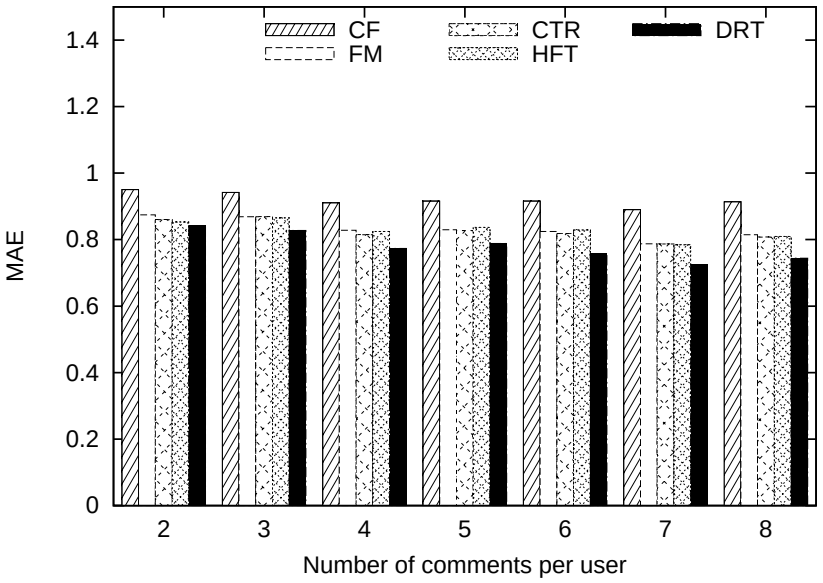

Figure 4: MAE vs number of training reviews per user (Amazon)

identify the feature words directly extracted from comments.

Recently some researchers try to learn latent aspects from comments (Wu and Ester, 2015). Wang et al. (2011) builds a regression model over the topic results from LDA. Their approach does not try to identify user profiles, and thus only applicable to score prediction on the text comments. Wang and Blei (2011) combine the merits of traditional collaborative filtering and topic modeling which provides a latent structure to recommend scientific articles in citation networks. Tang et al. (2015) learn representations of users and item for sentiment classification. Although similar concepts of user and item representations are used in (Tang et al., 2015), there are two major differences. Firstly, the problem we try to solve is completely different: they work on review sentiment classification, while our work focuses on score prediction without text. Secondly, the models are different: their approach includes user/product representation in CNN to enhance score prediction accuracy over text review, while out proposal attempts to maximize the likelihood of individual words. McAuley and Leskovec (2013) use a topic model to extract user and item profiles based on text comments. Although related, such approaches differ from ours in that they represent latent topics by keywords which are limited in representations. The approach of McAuley and Leskovec (2013) is the most relevant work to ours, but there are a number of limitations in their work. Firstly, they assume there is a one-to-one correspondence between the entries in the user profile and item profile. Secondly, their optimization relies on the assumption of LDA, such that the words are drawn independently from an unknown distribution. Our approach does not have any strong assumption on the generation mechanism.

6 Conclusion

In this paper, we have presented a simple yet effective approach to learn distributed representations of users and items from large amounts of text reviews. In our approach, the user vectors and item vectors are learned by a neural network. The user representation denotes the personal preferences over the items, while the item representation denotes the important attributes of the items. Finally, the distributed representations are used in recommendation systems. When tested on the datasets from different domains and languages, our systems achieve better performance than the state-of-the-art baseline systems.

Acknowledgments

Wenliang Chen, Zhenghua Li, and Min Zhang are supported by the National Natural Science Foundation of China (Grant No. 61572338, 61502325, and 61373095). This work is also partially supported by Collaborative Innovation Center of Novel Software Technology and Industrialization. Zhenjie Zhang

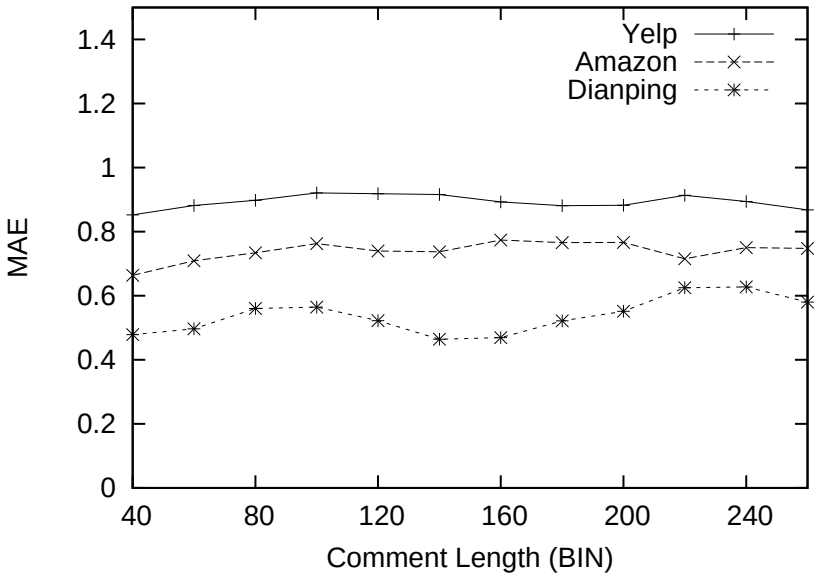

Figure 5: MAE vs average comment length (BIN) per user

is supported by the research grant for the Human Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores A*STAR. We would also thank the anonymous reviewers for their detailed comments, which have helped us to improve the quality of this work.

References

Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

David M Blei, T Griffiths, M Jordan, and J Tenenbaum. 2004. Hierarchical topic models and the nested chinese restaurant process. *NIPS*, 16:106–114.

Michael Gutmann and Anpo Hyvarinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistica. *The Journal of Machine Learning Research*, 13:307–361.

Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*.

Nitin Jindal and Bing Liu. 2008. Opinion spam and analysis. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 219–230. ACM.

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

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Bing Liu, Minqing Hu, and Junsheng Cheng. 2005. Opinion observer: analyzing and comparing opinions on the web. In *Proc of WWW*, pages 342–351. ACM.

Yue Lu, Malú Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *WWW*, pages 347–356.

Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*, pages 165–172.

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

Andriy Mhih and Hyee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of ICML*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Claudiu Cristian Musat, Yizhong Liang, and Boi Faltings. 2013. Recommendation using textual opinions. In *IJCAI*.

Steffen Rendle. 2012. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May.

Jason D. M. Rennie and Nathan Srebro. 2005. Fast maximum margin matrix factorization for collaborative prediction. In *ICML*, pages 713–719.

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML*, pages 880–887.

Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW Conference*, pages 285–295.

Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*, pages 650–658. ACM.

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

Duyu Tang, Bing Qin, and Ting Liu. 2015. Learning semantic representations of users and products for document level sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1014–1023, Beijing, China, July. Association for Computational Linguistics.

Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456. ACM.

Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *KDD*, pages 618–626.

Yao Wu and Martin Ester. 2015. Flame: A probabilistic model combining aspect based opinion mining and collaborative filtering. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 199–208. ACM.

Zhongwu Zhai, Bing Liu, Lei Zhang, Hua Xu, and Peifa Jia. 2011. Identifying evaluative sentences in online discussions. In *AAAI*.

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*, pages 83–92.

Rong Zhang, Zhenjie Zhang, Xiaofeng He, and Aoying Zhou. 2015. Dish comment summarization based on bilateral topic analysis. In *ICDE*, pages 483–494.

Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional matrix factorizations for cold-start recommendation. In *SIGIR*, pages 315–324. ACM.