Accepted Manuscript

Learning Representations from Heterogeneous Network for Sentiment Classification of Product Reviews

Lin Gui, Yu Zhou, Ruifeng Xu, Yulan He, Qin Lu

PII: S0950-7051(17)30114-4
DOI: 10.1016/j.knosys.2017.02.030
Reference: KNOSYS 3844

To appear in: *Knowledge-Based Systems*

Received date: 28 October 2016
Revised date: 26 February 2017
Accepted date: 27 February 2017

Please cite this article as: Lin Gui, Yu Zhou, Ruifeng Xu, Yulan He, Qin Lu, Learning Representations from Heterogeneous Network for Sentiment Classification of Product Reviews, *Knowledge-Based Systems* (2017), doi: 10.1016/j.knosys.2017.02.030

Learning Representations from Heterogeneous Network for Sentiment Classification of Product Reviews

Lin Gui[a,b], Yu Zhou[a], Ruifeng Xu[a,*], Yulan He[b], Qin Lu[c]

[a] *School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China*
[b] *School of Engineering and Applied Science, Aston University, United Kingdom*
[c] *Department of Computing, the Hong Kong Polytechnic University, Hong Kong*

Abstract

There have been increasing interests in natural language processing to explore effective methods in learning better representations of text for sentiment classification in product reviews. However, most existing methods do not consider subtle interplays among words appeared in review text, authors of reviews and products the reviews are associated with. In this paper, we make use of a heterogeneous network to model the shared polarity in product reviews and learn representations of users, products they commented on and words they used simultaneously. The basic idea is to first construct a heterogeneous network which links users, products, words appeared in product reviews, as well as the polarities of the words. Based on the constructed network, representations of nodes are learned using a network embedding method, which are subsequently incorporated into a convolutional neural network for sentiment analysis. Evaluations on the product reviews, including IMDB, Yelp 2013 and Yelp 2014 datasets, show that the proposed approach achieves the state-of-the-art performance.

Keywords: Sentiment Classification, Representation Learning, Network Embedding, Product Reviews

2010 MSC: 00-01, 99-00

☆ Fully documented templates are available in the elsarticle package on CTAN.

* Corresponding author

Email addresses: guilin.nlp@gmail.com; (Lin Gui), zhouyu.nlp@gmail.com; (Yu Zhou), xuruifeng@hitsz.edu.cn; (Ruifeng Xu), y.he9@aston.ac.uk; (Yulan He), csluqin@comp.polyu.edu.hk; (Qin Lu)

1. Introduction

Sentiment analysis [1, 2] becomes a hot topic in natural language processing studies. For a piece of text, a formal definition of sentiment analysis is given in [3] as a quadruple of (g, s, h, t). Here, g is an opinion target, s is the sentiment or the opinion about the target g, h is the opinion holder, and t is the $time$ when the opinion is expressed. The **opinion holder** usually refers to the person or organization who holds the opinion. The **opinion target** is the entity or a part or attribute of the entity that the sentiment or opinion has been expressed on.

In this paper, we focus on sentiment analysis in product reviews, in which the holder is usually the writer of a review and the target is a product. The key problem here is the detection of sentiment, a.k.a s in the aforementioned quadruple.

Traditionally, sentiment classification in product reviews is defined as a text classification problem at the document level. Supervised classifiers are trained from documents labeled with polarities and each document is represented using the bag-of-words assumption [4]. Other methods explored the use of latent topics for sentiment classification [5]. Most recently, there have been growing interests in using deep learning for sentiment classification. Different deep neural network architectures, such as Convolutional Neural Network [6, 7], Recursive Neural Network [8] or Recurrent Neural Network [9], have been explored to better represent sentences and documents which results in improved sentiment classification performance. Apart from text [10], user or product information can also be used in sentiment classification [11]. Such information can be represented as features to be combined with text features to train sentient classifiers.

Generally speaking, all the aforementioned methods learn a function that maps a given text into a certain sentiment class based on text representations or a combination of text and user/product representations. Learning of text representations is mostly done at the surface level of text using its lexical information. Text representation can be learned in different granularity levels. On the word level, the representation of a word is determined by its surrounding context. On the sentence or document level, the representation of text is derived by composing word-level representations in a sequential

order. Learning of user or product representations is independent from the learning of text representations. In this paper, we would like to explore a unified framework which can learn text and user/product representations simultaneously for sentiment classification.

In product reviews, we notice that users who share similar opinions to a certain product tend to use similar words in their comments. For example, fans of Sony camera may ridicule Zeiss using words such as "hammer". Users of Zeiss, on the other hand, describe Sony cameras as "toys", which implies that Sony cameras may look good but unprofessional. Words such as "hammer" or "toy" do not carry prior sentiments. However, they convey contextual sentiments in specific review articles. This observation inspires us to design a new method for representation learning by jointly consider words appeared in review text and its associated users and products.

In this study, we propose a framework to link users (opinion holders), reviews (that the users wrote) and products (opinion targets) through the words appeared in product reviews since the chosen words often reflect a user's viewpoint and the attributes of a product. More specifically, we propose an unified framework through the following two-level embedding learning based on a heterogeneous network. The bottom level is representation learning of words, users and products. More specifically, we link words with polarities, users and products based on the word statistics calculated from product reviews. Essentially, we have captured four types of information including word-word relations, word-user relations, word-target relations as well as word-polarity relations. Then, we use a network embedding method to identify similar words, users and products by mapping them into a unified continuous vector space. The continuous vectors (or representations) of words, users and products not only capture their contextual information, but also naturally embed their degrees of polarities. The top level is the representation learning of documents. We use the word representations learned from network embedding as the input to a convolutional neural network (CNN), and incorporate the learned user and product vectors as features into CNN for sentiment classification. Evaluations show that the proposed method achieves the state-of-the-art results on the IMDB, Yelp2013 and Yelp2014 datasets.

The main contributions of this paper are summarized below:

- A novel unified framework has been proposed to learn word, user and product representations simultaneously. To the best of our knowledge, this has never been explored before.
- The proposed network embedding learning method is able to map words, users and products into a unified continuous vector space to better capture the degree of polarities in words, users and products.
- The proposed framework for document-level sentiment classification has achieved the state-of-the-art performance on the IMDB, Yelp 2013 and Yelp 2014 datasets.

The rest of this paper is organized as following: Section 2 briefly introduces related work in sentiment analysis and discusses various modeling methods for users and products. Section 3 presents our approach including the construction of the heterogeneous network and a sentiment classifier based on CNN using the constructed network. Section 4 describes experimental setup and discusses performance evaluation results. Section 5 concludes the paper and outlines future research directions.

2. Related Work

In this section, we will present related work in sentiment classification with focus on learning word-level and sequence-level representations for sentiment classification. We will also briefly discuss existing approaches for user/product modeling.

2.1. Representation Learning for Sentiment Classification

Traditional machine learning methods to sentiment classification train supervised classifiers or regression models from text labeled with polarities [12]. Besides unigram word features, other features such as word n-grams, part-of-speech tags, negation words, modifiers, affective attributes, etc., are also used in sentiment classifier training [13, 14].

More recently, deep learning methods have been used in many text classification tasks including sentiment analysis [15, 16]. These methods aim to learn the continuous

representations of text, such as words, phrases, sentences and documents. Here, "continuous" follows the definition of "continuity" in mathematics. It requires the learning algorithm to map the words, phrases or sentences into a continuous vector which has continuous value in each dimension. Representation learning is typically carried out at two levels, namely the basic word level [17, 18] or the compositional sequence level [19, 20].

2.1.1. Word-level Representation Learning

The word-level representation, a.k.a word embedding, aims to map each word to a continuous low-dimensional space. The principle of mapping is to ensure the words sharing similar context should have similar representations in the low-dimensional space [17, 21]. Yogatama et al. [22] projected word embedding into a sparse vector. They found some linguistically interpretable dimensions. Faruqui and Dyer [23] used linguistic features to build word vector. Their results showed that these representations of word meanings can also achieve good performance in the analogy and similarity tasks. Qian et al. [24] proposed an algorithm to map different dense embeddings into a sparse linguistic property space. There are also some other methods for word embedding learning, such as the one proposed in [25], which used co-occurrence relations as a graph for word embedding learning.

Beside these word-level representation methods for universal NLP tasks, there are also some word embedding methods designed for sentiment analysis specifically. Tang et al. [18] considered the polarity of each sentiment word and learned a sentiment-specific word embedding for Twitter sentiment classification. They also used this method to built a large-scale sentiment lexicon for Twitter sentiment classification.

2.1.2. Sequence-level Representation Learning

The sequence level representation is designed to learn the continuous representation of a piece of text, such as a sentence, or a document. The learning algorithm for sequence level representation includes unsupervised methods [26] and supervised methods [8, 7]. Here, we focus on the supervised methods, which usually achieve better performance compared to unsupervised ones. There are two main types of learning

methods, the convolutional neural network (CNN) and the recurrent neural network (RNN). The former uses a convolutional operation to capture features from context to learn the representation of text [7]. The latter assumes that the representation of a word sequence is formed by the representations gradually built from the previous historical context. So, a recurrent mechanism is implemented to learn the representation of the whole word sequence [9]. There are also some modified methods proposed to learn the representation of text, such as using attention based weights to improve the performance [27], and a combination of RNN and CNN in different layers [9].

Nevertheless, to the best of our knowledge, there is no research which considered opinion holders and target simultaneously in word- or sequence-level embedding learning, even though such information constitutes an important part for sentiment analysis.

2.2. Sentiment Classification in Product Reviews

If we focus on the sentiment classification in product reviews, the methods can be categorized as text classification based methods and users/products modeling based methods.

2.2.1. Text Classification based Method

In principle, any text classification method can be implemented in sentiment classification. Training a classifier (such as SVM) with unigrams, bi-grams and trigrams as features is a strong baseline for sentiment classification[14]. Beside the text features above, the sentiment lexicon features or sentiment-specific word embeddings are also important in sentiment classification [18].

In recent years, the deep learning based method have achieved the state-of-the-art performance on this problem. Recursive Neural Tensor Network [8], Convolutional Neural Network [7] and Gated Recurrent Neural Network [9] achieved great success. The Recursive Neural Tensor Network [8] composes words into sentences by sharing parameters and syntactic structure. Convolutional Neural Network [7] uses convolutional filter to extract phrase level features, then implement a pooling operation to select the most relevant features to model a sentences. In Gated Recurrent Neural Network [9], sentences can be regard as a sequence of words, and any sequence can be

modeled by the last word and the previous sub sequence. These deep learning methods achieves state-of-the-art performance in different data set of sentiment classification task in product reviews.

However, the text classification based methods do not consider the user or product information, which is important for sentiment classification in product reviews.

2.2.2. User Modeling based Method

Apart from text, user information can also be used in sentiment classification. Gao et al. [28] designed user specific features to capture opinion holder leniency. Dong et al. [29] incorporated textual topics and user-word factors into supervised topic modeling. Hovy [30] used demographic information in sentiment analysis. Tan et al. [31] and Hu et al. [32] used user-user relationships for Twitter sentiment analysis.

Here, the deep learning method incorporating with user information achieves the best performance. For example, Tang et al. [33] incorporated the user and product information into convolutional neural networks for sentiment analysis. Recently, we also proposed a method to model users based on the concept of intersubjectivity [11]. The basic idea is to learn user embeddings based on their shared words in reviews which have similar polarities. However, what has been overlooked in these two methods is that the meanings of words may be also impacted by users and products (opinion targets) and hence representations of words should be updated together with user and product representations.

In summary, existing methods mainly considered three types of user information, namely: (1) personal profiles such as ages, gender, etc., to characterize users; (2) latent topics extracted from text as a proxy measure of users sharing similar topical interests; (3) rating patterns of users on product reviews. None of the above methods considered the subtle interplays between users and the words used by users sharing similar sentiments.

In this paper, we first propose a heterogeneous network embedding method to represent words, users and products in a unified embedding space based on the constructed heterogeneous network built on the word level. As will be shown in our experiments, three kinds of embeddings learned offer a better performance on the text classification

based method. The incorporating of user/product embeddings and convolutional neural network improves the user modeling based method and gives the state-of-the-art results on three product review datasets in document-level sentiment classification.

3. Our Approach

Our approach is inspired by the theory in sociology. In this section, we will first define our problem, and then discuss how to construct a heterogeneous network from text and other information. Next, we will present the network embedding method to model users, products and words in the same embedding space. Finally, we will explain how to incorporate the learned representations into a CNN for sentiment classification.

3.1. Problem Setup

In the corpora used in our experiments, we assume there are a total of D review articles written by $|U|$ users for $|P|$ products. Here, U and P stand for the set of users or products, $|\cdot|$ means the size of set. Each review article $x_i \in \mathbb{X}$ is represented by 3-tuple consisting of its author (or user) u_i, the opinion target (or product) p_i and the text content d_i, i.e.,

$$x_i = \{u_i, p_i, d_i\}, i \in \{1, 2, ..., D\} \tag{1}$$

Note that a user may post multiple reviews, and a product may receive reviews from different users.

We also assume that a review document d_i contains L_i words, $d_i = \{w_1^i, w_2^i, ..., w_{L_i}^i\}$, where w_j^i is the j-th word in document d_i. Given a fixed set of sentiment classes $\mathbb{Y} = \{y_1, y_2, ..., y_Y\}$, the goal of sentiment classification is to training a function F to map reviews to sentiment classes:

$$F : \mathbb{X} \to \mathbb{Y} \tag{2}$$

3.2. Word/User/Product Representation Learning with Network Embedding

For learning representations of words, users and products, it is important to define context for each of them. In traditionally word representation learning methods, such

as continuous bag-of-words (CBOW) or skip gram, the context for each word is typically defined as a 5-word window (two words before and after the target word). In our previously proposed user representation learning method [11], users should be similar to each other if they share similar subjective terms. Hence, the context for users is defined as their shared subjective terms in their reviews. In our work here, we argue that there exist subtle interplays among words, users and products. For example, words such as '*freezes*' and '*hangs*' are used often in negative reviews towards mobile phones. These two words should carry similar semantic meanings and hence their representations should be placed in nearby locations in the embedding space. As such, word, user and product representation learning should be performed simultaneously to map them into a unified embedding space.

We propose to first build a heterogeneous network, in which words, users, products, as well as sentiment labels are vertices and statistical relations between them are edges. Then we use a network embedding method to learn the distributed representation of each vertex including words, users and products.

3.2.1. Construction of Heterogeneous Network

Here, we define the network as a graph: $G = \{E, V\}$. The E is the set of edges and V is the set of vertices, which is the union of all words, users, and products. We need to learn the weight of each edge, define as $\omega(e), e = \{u, v\}, e \in E$ and $u, v \in V$ here.

In order to capture the information of users and products, we have included an additional type of vertices, polarities. Four categories of relations should be considered in the construction of heterogeneous network (Fig. 1):

- Word-word relation (Figure. 1A). If two words appear in a context within a window of size k, we consider them to have a word-word relation. In this study, k is 5 and the weight of edges, which indicate the word-word relation, is based on the frequency of occurrences in the same context;

- Word-polarity relation (Figure. 1B). In order to capture the polarity of a word, we define the weight of word-polarity relation as how many times the word appeared

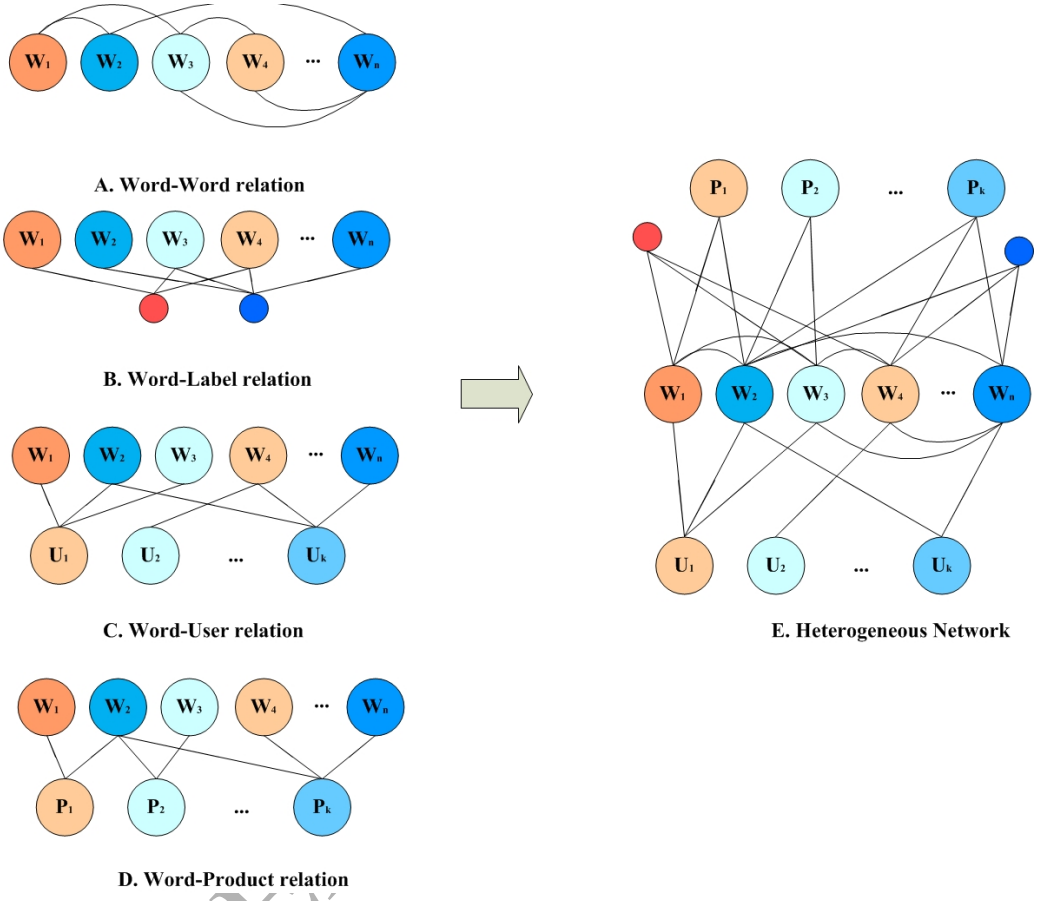


Figure 1: Heterogeneous network.

in a review document with a specific polarity label;

- Word-user relation (Figure. 1C). The weight of the edge of word-user relation is simply defined as how many times the word is used by the user in his/her reviews;

- Word-product relation (Figure. 1D). We define the weight of the edge of word-product relation in a similar way to the word-user relation.

Based on these four types of relations defined above, we can obtain a heterogeneous

Algorithm 1 Heterogeneous network construction.

Input: labeled review documents $(x_i, y_i), x_i = \{u_i, p_i, d_i\}, i \in \{1, 2, ..., N\}$ for $|U|$ users and $|P|$ products; $d_i = \{w_1^i, w_2^i, ..., w_{L_i}^i\}$
Initialization: $G = \{E, V\}$, $V = \{u_1, ..., u_{|U|}, p_1, ..., p_{|P|}, w_1^i, ...w_{L_i}^i\}, i \in \{1, 2, ..., N\}$, initialize the weight of each pair of vertices $\omega(v_i, v_i') = 0$.
for all $x_i = \{u_i, p_i, d_i\}, i \in \{1, 2, ..., N\}$ **do**
    for all $w_{ij}, j = 1, 2, ..., L_i$ **do**
        $\omega(u_i, w_{ij}) + +$
        $\omega(p_i, w_{ij}) + +$
        $\omega(y_i, w_{ij}) + +$
        for all $w_{ik}, k = 1, 2, ..., L_i$ **do**
            if $k - j > 0$ && $k - j \leq 2$, **then**
            $\omega(w_{ik}, w_{ij}) + +$
        end for
    end for
end for
Output: $G = \{E, V\}$ with weights of each paris of vertices $\omega(v_i, v_i')$ updated.

network (Figure. 1E). Here, the heterogeneous network is an undirected graph. The pseudo-code of the heterogeneous network construction details is given in Algorithm 1. The next problem is how to learn the embedding for each vertex in the network.

3.2.2. Network Embedding for Representation Learning

We propose to learn node representations in the heterogeneous network using network embedding [33]. Given a large network $G = (E, V)$, where V is a set of vertices, and E is a set of edges, which stands for the relationship between vertices. Network Embedding aims to represent each vertex $v \in V$ in a lower-dimensional space $\mathbb{R}^d$ by learning a function $f : V \to \mathbb{R}^d$, where $d \ll |V|$. Here, $\mathbb{R}^d$ means an Euclidean d-spaces.

In this method, we assume that each vertex in the network has a corresponding vector, or representation. Then, we can use the representations of nodes to calculate

the weight on the edges. The basic idea is to define a cost to measure the distinction between the calculated result and the real weight in the network. Then, use gradient descent method to achieve the minimum cost then we can get the representations of the nodes. It is a feature extraction method for the network, and we can use this features to reconstruct the network obviously. The details of algorithm is given in Algorithm 2:

Here, for any vertex v_i and its corresponding representation α_i in the vector space, where $\alpha_i \in \mathbb{R}^d$, let the neighbor of v_i in the heterogeneous network be denoted as v_i' with the representation α_i'. The conditional probability $p(v_j|v_i)$, which means that the probability of v_j is neighbor of v_i in the network, or the weight on the edges, can be defined by softmax as:

$$p\left(v_j|v_i\right) = \frac{\exp\left({\alpha'}_j^T \alpha_i\right)}{\sum_{k=1}^{|V|} \exp\left({\alpha'}_k^T \alpha_i\right)} \tag{3}$$

In order to preserve the distribution of edges, we should make the conditional distribution of $p(v_j|v_i)$ defined in Equation 3 to be close to its empirical distribution $\omega(v_j, v_i)$, which is a statistical result from the network. Here, we hope that the empirical distribution can be derived from weights in the heterogeneous network. We define the objective function O of our representation learning algorithm as:

$$O = -\sum_{(v_i,v_j)\in E} \omega(v_j, v_i) \log p\left(v_j|v_i\right) \tag{4}$$

Here, $\omega(v_j, v_i)$ is a statistical result. So, it is fixed in our algorithm. Then, the objective function (4) can be considered as inner product of representation based probability and $\omega(v_j, v_i)$. Since the larger inner product indicates higher similarity. So, by minimizing the objective function (4), we can represent every v_i with a d-dimensional vector α_i such that the defined probability is similar with the statistical result. This way, words with the same polarity and used on the same product will be similar to each other in the low dimensional representation space, and the users/product sharing similar terms will be similar to each other too.

In the learning of network embedding, the optimization of objective function is based on negative sampling. Since our network is heterogeneous and contains four

Algorithm 2 Network embedding learning for generating vertices' representations.

Input: $G = \{E, V\}, E = \{E_U \cup E_P \cup E_W \cup E_Y\}, V = \{U \cup P \cup W \cup Y\}$, number of vertices T, number of negative samples Q

Initialization: for each vertex $v_i \in V$, randomly initialize its representation α_i

Tranining:

while Number of iterations $< T$ **do**

  Stochastic choosing $E_S \in \{E_U, E_P, E_W, E_Y\}$

  Sample an edge from E_S

  Draw Q negative edges

  Minimize the objective function (4)

  Update the vertices' representations

end while

Output: word, user and product representations

types of different relations, we need to ensure the sampling for each relation type is equally probable. Thus, we stochastically choose a relation type at each iteration before sampling an edge of this type. Our adapted network embedding learning method is shown in Algorithm 2:

Since $\omega(v_j, v_i)$ calculates the frequency of context, this objective function is actually similar to the skip-gram model and continuous bag of words (CBOW) model [17]. The skip-gram model predicts the context by given words and the CBOW model predicts the word by given context. Our model can be considered as a combination of skip-gram and CBOW if we only consider word-word relations in Algorithm 1 since our network is an undirected graph. In these two models, the object function which only considers the word-word relation is defined as:

$$O_{word2vec} = -\sum_{v_i \in \text{ document}} \left(\sum_{v_j \in \text{ context}} \log p\left(v_j | v_i\right) \right) \tag{5}$$

The right-hand-side of equation(5) is calculated based on the number of co-occurrence of v_i and v_j in the same context window. Since we also use the co-occurrence frequency as the weight of the link connecting two words in the construction of heteroge-

neous network in Algorithm 1, for a word v_i, it can be easily seen that:

$$-\sum_{v_j \in \text{ context of } v_i} \log p\left(v_j|v_i\right) = -\sum_{(v_i,v_j)\in E} \omega(v_j,v_i)\log p\left(v_j|v_i\right) \tag{6}$$

Then, we can obtain the following equation:

$$\begin{aligned} O &= -\sum_{v_i \in \text{ document}} \left(\sum_{v_j \in \text{ context}} \log p\left(v_j|v_i\right)\right) \\ &= -\sum_{(v_i,v_j)\in E} \omega(v_j,v_i)\log p\left(v_j|v_i\right) \\ &= O_{word2vec} \end{aligned}$$

Thus, if we only consider the word-word relation in the heterogeneous network, our objective function (4) is equivalent to (5) in the skip-gram model and CBOW model. The difference is that with our heterogeneous network, the context of a word contains much richer information as it also includes users who used it, the products that it is associated with and the polarity of the review documents where the word occurred. As such, the word embeddings learned by our method can effectively capture the user and product information to a certain degree. Moreover, user and product representations can be obtained in the the same learning process. As will be shown later in our experiments, the more contextual information is added, the better embeddings are leaned. And the user/product representations can be added as additional features to improve the performance of sentiment classification.

3.3. Incorporating Vertices' Representations into CNN for Sentiment Classification

Once word, user and product representations are learned, we show in this section how these representations can be incorporated into a convolutional neural network [7] for document-level sentiment classification. The architecture of the neural network is shown in Figure 3. The input layer is the continuous word representations learned by the network embedding method mentioned in Section 3.2.2.

Assuming that the i-th review document $d_i = \left\{w_1^i, w_2^i, \ldots, w_{L_i}^i\right\}$ contains L_i words with a sentiment label y_i, written by user u_i about product p_i, we have a labeled input (x_i, y_i) where $x_i = \{d_i, u_i, p_i\}$. Assuming each word w_j in document d_i

is represented by a vector with n dimensions, denoting the concatenation of a word sequence from the j-th word to the k-th word as $w^i_{j:k}$, a convolution operation involving a filter $m \in \mathbb{R}^{h \times n}$, which have the window of size h, can extract a feature on $w^i_{j:j+h-1}$ in the convolutional layer by:

$$c^{im}_j = f\left(x \cdot w^i_{j:j+h-1} + b\right) \tag{7}$$

Here, b is the bias, x is the parameter in convolutional filter, and f is a non-linear mapping function such as the logistic function. In this equation, we use the inner product of filter and word sequence as an input of mapping function. The inner product actually indicate the similarity between the filter parameter and word sequence. The mapping function is monotonous and bounded. So, if the word sequence is similar with the filter, there will be a positive output after mapping. Each filter in our algorithm is associated with a label. In another word, the positive mapping result means that the word sequence can be used as features for this label.

We then use Equation (7) to extract the features from document d_i as:

$$\left(c^{im}_1, c^{im}_2, c^{im}_3, \ldots, c^{im}_{l-h+1}\right) \tag{8}$$

A max pooling operation is deployed to extract the most relevant feature, which has the highest value in the max pooling layer:

$$\hat{c}^{im} = \max_{i=1,2,\ldots,l-h+1} \left(c^{im}_1, c^{im}_2, c^{im}_3, \ldots, c^{im}_{l-h+1}\right) \tag{9}$$

The convolutional operation can be repeated several times with different initialized filters in order to extract different features. Finally, all the features are concatenated as the representation of the document, denoted as α_{d_i}:

$$\alpha_{d_i} = \{\hat{c}^{i1}, \hat{c}^{i2}, \ldots \hat{c}^{im}\}, \tag{10}$$

In order to compose with the user/product representations learned from heterogeneous network embedding results, we concatenate the user embeddings and product embeddings with the document representation in the max pooling layer, i.e., $\alpha_{d_i} \bigoplus \alpha_{u_i} \bigoplus \alpha_{p_i}$. Then, a softmax layer is added to train a classifier for sentiment classification.

In the pre-training, the corpora is tokenized and all tokens have been kept. In the training, weights associated with both heterogeneous network embedding features and the convolutional features are updated simultaneously. The dimensions of the embedding space is 300, the window sizes of convolutional filters are 2, 3, 4, and 5. For each window size, the number of convolutional filters is 100. So, there are 400 features after the max pooling operation. The mini-batch of training is 100 and we use the stochastic gradient descent as the optimization method.

4. Experiments

4.1. Experimental Setup

We evaluate our algorithm on three product review datasets including IMDB [34] and the Yelp Dataset Challenge both in 2013 and 2014. Statistics with respect to users, products and reviews are given in Table 1:

Dataset	#Class	#Users	#Products	#Reviews	$\|V\|$	Length
IMDB	10	1,310	1,635	84,919	91,808	395.21
Yelp2013	5	16,31	1,633	78,966	96,817	188.76
Yelp2014	5	4,818	4,194	231,163	183,541	196.06

Table 1: The statistics of three datasets. (Here, $|V|$ is the size of vocabulary, Length is the #words per review.)

IMDB is labeled with 10 different levels of sentiment, score 1 for the most negative and score 10 for the most positive. Yelp 2013 and Yelp 2014 are labeled with 5 different levels of sentiment, score 1 for the most negative and score 5 for the most positive.

We use accuracy (ACC) as the evaluation metrics. Let the predicted label of i-th testing sample be $predicted_i$, the actual label of i-th testing sample be $actual_i$, and the size of the test set be N, ACC is calculated by:

$$\text{ACC} = \frac{\sum_{predicted_i = actual_i} 1}{N} \tag{11}$$

4.2. Baselines

We compare our proposed method with a number of existing methods as listed below:

- **Majority** simply takes the most prominent sentiment category in the training set as the sentiment label of each document in the test set.
- **Trigram** trains a Support Vector Machine (SVM) classifier with unigrams, bigrams and trigrams as features.
- **TextFeature** extracts text features including word and character n-grams, sentiment lexicon features, etc., and then train a SVM classifier.
- **UPF** extracts user-leniency features [28] and the corresponding product features from the training data, which is further concatenated with the features in Trigram an TextFeature.
- **AvgWordvec** averages word embeddings in a document to obtain the document representation which is fed into a SVM classifier as features.
- **SSWE** generates features with sentiment-specific word embeddings (SSWE) [18] and then trains a SVM classifier.
- **Paragraph Vector** for document modeling [26];
- **RNTN** Recursive Neural Tensor Network for sentence modeling [8] which is incorporated into recurrent neural networks for document modeling;
- **CNN** Convolutional Neural Network for sentiment classification [7];
- **GRNN** Document modeling with gated recurrent neural network for sentiment classification [9];
- **UPNN** User Product Neural Network [33], which incorporates user and product information using CNN;
- **ISN** Intersubjectivity Network Embedding [11], which incorporates user representation learned from intersubjectivity network with a CNN based text classification.

4.3. Overall Results

Methods	IMDB	Yelp2013	Yelp2014
Majority	0.196	0.411	0.392
Trigram	0.399	0.569	0.577
Textfeature	0.402	0.556	0.572
AvgWordvec+SVM	0.304	0.526	0.530
SSWE+SVM	0.312	0.549	0.557
Paragraph Vector	0.341	0.554	0.564
RNTN	0.400	0.574	0.582
UPNN (CNN without user&product features)	0.405	0.577	0.585
GRNN	0.443	0.614	0.621
Our approach (without user&product features)	**0.488**	**0.623**	**0.637**

Table 2: Comparison with existing methods without user and product information.

We first conduct experiments without the use of user and product features and show the results in Table 2. Here, Trigram and Textfeature are traditional classification methods. The AvgWordvec+SVM and SSWE+SVM incorporate traditional methods with deep learning based features, such as word representation. It is obviously that using deep learning results as features without composition methods, such as CNN or RNN, can not improve the performance of sentiment classification. The Paragraph Vector, RNTN, UPNN and GRNN are deep learning methods. Here, Paragraph Vector is unsupervised method, so the performance is lower than other three methods. RNTN, UPNN and GRNN use composition of word representation to obtain sentence representation or document representation.

It can be observed that our method significantly outperforms all the other baselines including those neural network models specifically designed for sentiment classification, such as the hierarchical gated RNN for document modeling (GRNN) [9]. Also, we notice that UPNN without user and product information is actually a CNN trained from word embeddings. The main difference between our method and UPNN is the word representations derived. We generate word representations from our constructed

heterogeneous network, which inherently considers the user, product and polarity information during representation learning. On the contrary, UPNN only used word embeddings trained with context information from text content only. The results shown in Table 2 demonstrate that opinion holder and target information can be crucial for learning word embeddings specifically for sentiment analysis.

Methods	IMDB	Yelp2013	Yelp2014
Trigram+UPF	0.404	0.570	0.576
TextFeature+UPF	0.402	0.561	0.579
UPNN	0.435	0.596	0.608
ISN	0.476	0.623	0.635
Our approach (with user&product features)	**0.509**	**0.656**	**0.662**

Table 3: Comparison with existing methods with user and product information in terms of ACC.

We then conduct another set of experiments by incorporating user and product information for sentiment classification. We only include the results from those baselines where it is possible to add in user and product features. The results are shown in Table 3. It can be observed that simply adding user and product information into text features for SVM training gives improved performance (Trigram+UPF and TextFeature+UPF). UPNN derived user and product representations based on the review rating information while ISN learned user representations based on shared subjective terms. The results show that the latter appears to be more effective compared to the former. Nevertheless, our proposed method outperforms the previous state of the art system, ISN, by 3.2% on average with p-value< 0.01 which indicates a significant improvement.

4.4. Learning with Different Embeddings and Features

In order to examine the effect of using different embeddings and learning representations using different sets of information, we use CNN as the base model and experiment with different embeddings. The variants we have evaluated are listed below:

- **CNN+word2vec** uses the word embeddings trained using word2vec as the input to CNN;

- **CNN+W** uses the word-word and the word-polarity relations to learn word representations in our proposed architecture for CNN;
- **CNN+WU** adds the word-user information in addition to word-word and the word-polarity relations for learning word representations. Note that even though user representations have also been generated, they are not used in classification;
- **CNN+WP** adds the word-product information in addition to word-word and the word-polarity relations for learning word representations. Similar to CNN+WU, the product information is only used to updated word representations and it is not used in classification;
- **CNN+WUP** combines all the four types of relations, word-word, word-polarity, word-user and word-product, for word representation learning. Again, only word representations are used in classification;
- **CNN+WUP+V(U)** similar to CNN+WUP except that the user representations are concatenated with the document-level features captured at the max pooling layer;
- **CNN+WUP+V(P)** similar to CNN+WUP+V(U) except that the product representations are used instead of user representations;
- **CNN+WUP+V(UP)** refers to the complete architecture we proposed in this paper.

It can be observed from Table 4 that learning word representations with the document polarity label information (CNN+W) significantly improves the sentiment classification by 8% in accuracy on average compared to simply using word embeddings trained from a large raw text corpus (CNN+word2vec). Adding user and product information separately or simultaneously for word representation learning gives mixed results. Compared to CNN+W, we observe improved accuracies on IMDB and Yelp2014, but either no change or a slight drop in accuracy on Yelp2013. However, if the learned user and product representations are concatenated with document representations for

Methods	IMDB	Yelp2013	Yelp2014
CNN+word2vec	0.412	0.532	0.520
CNN+W	0.472	0.628	0.607
CNN+WU	0.481	0.625	0.610
CNN+WP	0.484	0.628	0.612
CNN+WUP	0.488	0.623	0.637
CNN+WUP+V(U)	0.502	0.641	0.649
CNN+WUP+V(P)	0.502	0.633	0.644
CNN+WUP+V(UP)	0.509	0.656	0.662

Table 4: Comparison with different embeddings and features.

CNN training, we see the performance improvement over CNN+W in the range of 1.3-4.2%. Our full architecture (CNN+WUP+V(UP)) gives the best results overall.

4.5. Further Analysis on Embeddings

In order to gain a better understanding of the our network embedding results, we make a further analysis on embeddings in this section. The goal of embedding is to map the nodes into a continuous vector space which preserves the properties of input network. As such, we seek for answers to the following two questions:

- For the positive/negative users/products/words in the heterogeneous network, will they preserve their polarities in the embedding space?
- For the positive/negative users/products/words in the embedding result, what are their original distributions in the heterogeneous network?

4.5.1. Polarity in the Embedding Space

For the first issue, we plot in Figure 3 the learned users, products and words embeddings in our heterogeneous network. Here, we use Principal Component Analysis (PCA) to generate the top two dimensions of the heterogeneous network embedding results in the three training datasets. Specifically, for the Yelp2013/2014 datasets which

have the ratings in the scale of 1 to 5, we mark users or products as positive if their average ratings are 4 or above, and negative if their average ratings are below 3. For the IMDB dataset which have ratings in the scale of 1 to 10, we mark the users or products by using the positive rating threshold of 6 and negative rating threshold of 5. For plotting words embedding, we use the log ratio method, which has been previously used to extract the sentiment features in sentence classification [35], to select words. If we denote the weight between word j and positive/negative node as $Positive_j$ / $Negative_j$, then we can define the vectors as: $p_j = \beta + Positive_j$, and $q_j = \beta + Negative_j$ where β is a smoothing parameter. In our experiment, we take β as 0.05. The log ratio of word j is:

$$ratio_j = \log(\frac{p_j / \parallel p \parallel^1}{q_j / \parallel q \parallel^1}). \tag{12}$$

We plot in Figure 3 the top 50 positive and negative users/products/words based on their respective statistical information aforementioned above in our heterogeneous network. It can be observed that in general users, products or words with different polarities are separated well. We also notice that there are some overlapping positive/negative users and products as shown in Figure 3(a), (d) and (e). This is because we only use a simple thresholding method to mark users and products with different colors for a better visualization. However, it might be the case that mixed reviews containing more positive comments were assigned with lower rating scores. Hence, the document-level rating scores do not always reflect the actual degrees of positivity or negativity in review content. We argue that using words to learn user and product embeddings would give more accurate results compared to using rating scores. Overall, we can see that word/user/product embeddings learned using our proposed network embedding method indeed make sense.

4.5.2. Distribution in heterogeneous Network

For the second issue, we next explore in details the top 5 most positive and negative users identified by our heterogeneous network embedding from the three datasets. Recall that we have four types of nodes in the heterogeneous network: the user nodes, the term nodes, the product nodes, and the polarity nodes. We use the similarity between

the user nodes and polarity nodes in the embedding space to identify the top 5 most positive and most negative users. The results are shown in Table 5 (a)-(b) (IMDB), Table 6 (a)-(b) (Yelp2013) and Table 7 (a)-(b) (Yelp2014). We notice that these users all share very similar review patterns. For example, users in the top 5 most positive group tend to give high ratings (most of the time above 7 for IMDB and above 4 for Yelp datasets) in most of their reviews, while users in the top 5 negative group tend to give low ratings (most of the time below 5 for IMDB and below 3 for Yelp datasets). In a similar way, we also list the top 5 most positive and most negative products in Table 5 (c)-(d) (IMDB), Table 6 (c)-(d) (Yelp2013) and Table 7 (c)-(d) (Yelp2014). We observe a similar rating patterns for products compared to those given by users.

It is worth noting that the similarity values between user/product nodes and polarity nodes in our heterogeneous network are not directly related to the average rating scores of users/products since we learn user or product embeddings based on the words occurred in their associated reviews. We can see from Table 6(a) that the user who is ranked in the third place only has an average rating score of 3.66, but he is among the top 5 most positive users in the Yelp 2013 dataset. By closely examining his reviews, we find that he tends to give very positive comments even in reviews with the rating of 3. For example:

"This is a cool place. The sauces are great, my fav is the devils spit and texas pit, but the BBQ is just OK..."

"Yum. I with I had learned of this place sooner everything we ordered was delicious. The raspberry margarita was so good and refreshing..."

"Great place to watch a game with friends and have a few beers. Its nice and clean with lots of TVs and seating..."

In a similar way, we also list the top 30 most positive and most negative words in Table 8. It can be observed that the polarity words identified are context-dependent. For example, we found a negative word "*magorium*" in the IMDB dataset. It is actually the name of a film which only has an average score of 6.2, showing that it received more negative reviews compared to positive ones. Yelp 2013 and 2014 datasets share many top positive and negative words. This is not surprising since they contain reviews towards similar products but posted in different years.

(a) Top 5 most positive users.

Similarity	Review history(times)										Average
	1	2	3	4	5	6	7	8	9	10	
0.699	0	0	0	0	1	2	3	9	18	14	8.76
0.683	0	1	1	2	3	7	10	16	9	30	8.21
0.674	1	1	0	0	3	0	0	5	4	10	8.08
0.631	0	1	1	1	6	5	22	18	21	16	7.86
0.629	3	2	3	0	4	2	13	12	15	31	8.00

(b) Top 5 most negative users.

Similarity	Review history(times)										Average
	1	2	3	4	5	6	7	8	9	10	
0.665	0	6	8	5	7	4	2	1	0	0	4.15
0.659	13	5	1	0	2	0	1	2	1	8	4.48
0.655	13	3	13	10	11	7	7	7	1	6	4.73
0.648	13	25	20	20	17	22	9	13	7	7	4.70
0.641	14	8	10	10	3	4	7	2	0	2	3.73

(c) Top 5 most positive products.

Similarity	Review history(times)										Average
	1	2	3	4	5	6	7	8	9	10	
0.711	3	0	1	0	0	5	6	13	18	116	9.28
0.649	0	0	0	0	2	1	6	25	26	41	8.93
0.622	1	0	0	0	0	4	3	11	28	62	9.22
0.613	3	3	3	3	6	7	10	15	22	74	8.42
0.609	3	0	1	0	0	5	6	13	18	116	9.28

(d) Top 5 most negative products.

Similarity	Review history(times)										Average
	1	2	3	4	5	6	7	8	9	10	
0.679	13	9	7	6	6	5	4	3	1	2	3.87
0.666	10	11	7	10	4	3	5	7	0	4	4.29
0.654	10	6	9	6	9	3	1	1	0	2	3.63
0.651	11	4	3	3	1	1	2	1	1	1	3.28
0.650	12	8	10	7	11	4	1	0	0	2	3.49

Table 5: The most positive/negative users/products identified by embeddings in IMDB.

4.6. Future work

From aforementioned results, we know that the proposed method is reasonable and achieves better performance than existing methods. In the future, we can apply our method in social analysis. In our paper, we use comments to model users/product, and update word embeddings at the same time. However, the same words for different

(a) Top 5 most positive users.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.681	1	2	4	9	22	4.28
0.672	1	4	3	5	19	4.16
0.669	2	2	13	31	5	3.66
0.668	0	0	2	12	12	4.38
0.662	0	1	2	7	6	4.13

(b) Top 5 most negative users.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.711	11	4	1	8	4	2.64
0.684	4	5	1	4	2	2.69
0.675	2	7	2	2	3	2.81
0.623	2	4	10	10	2	3.21
0.611	2	5	1	8	0	2.93

(c) Top 5 most positive products.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.669	0	3	6	7	15	4.09
0.651	0	1	9	21	12	4.02
0.644	0	4	4	12	7	3.81
0.617	1	5	10	28	24	4.01
0.599	1	3	12	42	22	4.01

(d) Top 5 most negative products.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.712	6	3	7	6	3	2.88
0.708	8	7	8	3	1	2.33
0.659	6	7	5	3	0	2.23
0.655	9	4	4	9	1	2.59
0.631	6	12	27	19	6	3.10

Table 6: The most positive/negative users/products identified by embeddings in Yelp2013.

(a) Top 5 most positive users.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.699	0	1	2	10	11	4.29
0.683	0	0	4	2	10	4.37
0.674	2	1	2	7	12	4.08
0.631	0	0	3	4	28	4.71
0.629	0	1	2	16	19	4.39

(b) Top 5 most negative users.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.665	4	2	3	5	2	2.93
0.659	7	6	20	21	2	3.08
0.655	7	4	2	4	3	2.60
0.648	8	3	1	3	5	2.70
0.641	7	4	3	4	4	2.72

(c) Top 5 most positive products.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.711	0	3	7	25	33	4.29
0.649	0	1	4	10	5	3.95
0.622	1	1	1	8	12	4.40
0.613	1	0	3	5	14	4.34
0.609	0	1	9	32	12	4.01

(d) Top 5 most negative products.

Similarity	Review history(times)					Average
	1	2	3	4	5	
0.679	5	10	5	1	0	2.09
0.666	18	6	4	2	0	1.66
0.654	4	3	4	6	1	2.83
0.651	11	11	26	22	8	3.06
0.650	4	6	6	7	1	2.79

Table 7: The most positive/negative users/products identified by embeddings in Yelp2014.

latent communities may have different meanings. our method can be used to learn multi-prototype word embeddings to identify different latent communities.

Another future work is to apply user/product embedding in sentiment analysis. In

Dataset	Top 30 negative words	Top 30 positive words
IMDB	travesty, worst, crap, stupid, olds, rotten, pointless, awful, gervaise, dreadful, pitiful, miserable, disgusting, sucks, retarded, idiotic, waste, psychlos, magorium, drivel, pathetic, stinking, vapid, junk, ho, insulting, trash, atrocious, uwe, sickening	decent, cast, life, nice, kind, great, long, young, times, worth, interesting, story, special, fact, father, left, comedy, scenes, family, original, scene, action, high, love, guy, girl, day, moments, pretty, funny
Yelp2013	horrible, stale, unhappy, worst, flies, worse, mediocre, blatantly, bland, disappointment, fail, awful, disaster, poorly, downhill, refused, miserable, lousy, disgusting, sucks, nasty, annoyed, terrible, hopes, waste, poor, incompetent, disappointing, pathetic, uncomfortable	awesome, nice, loved, favorite, great, perfect, cool, worth, special, red, fresh, lot, home, top, find, flavor, fantastic, excellent, love, bread, sweet, pretty, big, wonderful, huge, fun, atmosphere, staff, super, enjoy
Yelp2014	horrible, stale, unhappy, worst, worse, mediocre, bland, acted, rotten, arguing, disappointment, awful, disaster, poorly, fraud, downhill, refused, miserable, lousy, disgusting, sucks, nasty, annoyed, terrible, crappy, FALSE, waste, poor, incompetent, disappointing	awesome, nice, loved, favorite, great, perfect, bit, recommend, worth, special, fresh, top, ice, flavor, feel, experience, friendly, happy, amazing, good, hot, cream, selection, free, delicious, sandwich, tasty, dessert, pork, day

Table 8: The most positive/negative words identified by embeddings.

this work, our algorithm learns user/product embedding. However, it is too simple to use them merely as features in classification. In the future, we will consider to stack attention mechanism on a deep network architecture to capture user/product information by learned embeddings.

5. Conclusion

In this paper, we present a new method for sentiment classification on the product reviews. It is based on the notion of heterogeneous network representation. More specifically, we include users (opinion holders), words, products (opinion targets) and polarities in a unified framework. Words, users and products are all mapped into the same embedding space. The learned user and product representations are then incorporated into a CNN-based neural network for sentiment classification. Experimental results show that our proposed embedding learning method not only offers a better semantic interpretation incorporating user and product information but also improved sentiment classification to achieve the state-of-the-art results on the relevant datasets.

Acknowledgements

This work was supported by the National Natural Science Foundation of China 61370165, U1636103, 61632011, National 863 Program of China 2015AA015405, Shenzhen Foundational Research Funding JCYJ20150625142543470 and Guangdong Provincial Engineering Technology Research Center for Data Science 2016KF09.

References

[1] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up? sentiment classification using machine learning techniques, Computer Science, 2009, pp. 79–86.

[2] R. Xu, L. Gui, J. Xu, Q. Lu, K. Wong, Cross lingual opinion holder extraction based on multi-kernel svms and transfer learning, World Wide Web 18 (2), 2015, pp. 299–316.

[3] B. Liu, Sentiment analysis and opinion mining, Vol. 5, Morgan & Claypool Publishers, 2012.

[4] L. Gui, R. Xu, Q. Lu, J. Xu, J. Xu, B. Liu, X. Wang, Cross-lingual opinion analysis via negative transfer detection, in: Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, USA, 2014, pp. 860–865.

[5] Y. He, C. Lin, W. Gao, K. F. Wong, Dynamic joint sentiment-topic model, Acm Transactions on Intelligent Systems and Technology 5 (1), 2013, pp. 102–114.

[6] B. Hu, Z. Lu, H. Li, Q. Chen, Convolutional neural network architectures for matching natural language sentences, in: Annual Conference on Neural Information Processing Systems, Montréal, Quebec, Canada, 2014, pp. 2042–2050.

[7] Y. Kim, Convolutional neural networks for sentence classification, Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 2014, pp. 1746–1751.

[8] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, Conference on Empirical Methods in Natural Language Processing, Seattle, USA, 2013, pp. 1631–1642.

[9] D. Tang, B. Qin, T. Liu, Document modeling with gated recurrent neural network for sentiment classification, in: Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 2015, pp. 1422–1432.

[10] R. Dunbar, Grooming, gossip, and the evolution of language. Harvard University Press, 1998.

[11] L. Gui, R. Xu, Y. He, Q. Lu, Z. Wei, Intersubjectivity and sentiment: from language to knowledge, in: International Joint Conference on Artificial Intelligence, New York, USA, 2016, pp. 2789–2795.

[12] B. Pang, L. Lee, Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales, in: Annual Meeting of the Association for Computational Linguistics, University of Michigan, USA, 2005, pp. 115–124.

[13] A. Abbasi, S. France, Z. Zhang, H. Chen, Selecting attributes for sentiment classification using feature relation networks, IEEE Transactions on Knowledge and Data Engineering 23 (3), 2011, pp. 447–462.

[14] R. Xia, C. Zong, X. Hu, E. Cambria, Feature ensemble plus sample selection: Domain adaptation for sentiment classification, Intelligent Systems IEEE 28 (3), 2013, pp. 10–18.

[15] D. Tang, F. Wei, B. Qin, et al., Building large-scale twitter-specific sentiment lexicon: A representation learning approach, in: International Conference on Computational Linguistics, Dublin, Ireland, 2014, pp. 172–182.

[16] L. Xu, K. Liu, J. Zhao, Joint opinion relation detection using one-class deep neural network, in: International Conference on Computational Linguistics, Dublin, Ireland, 2014, pp. 677–687.

[17] T. Mikolov, I. Sutskever, K. Chen, et al., Distributed representations of words and phrases and their compositionality, in: Annual Conference on Neural Information Processing Systems, Lake Tahoe, Nevada, United States, 2013, pp. 3111–3119.

[18] D. Tang, F. Wei, N. Yang, et al., Learning sentiment-specific word embedding for twitter sentiment classification, in: Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, USA, 2014, pp. 1555–1565.

[19] X. Glorot, A. Bordes, Y. Bengio, Domain adaptation for large-scale sentiment classification: A deep learning approach, in: International Conference on Machine Learning, Bellevue, Washington, USA, 2011, pp. 513–520.

[20] N. Kalchbrenner, E. Grefenstette, P. Blunsom, A convolutional neural network for modelling sentences, in: Annual Meeting of the Association for Computational Linguistics, Baltimore, Maryland, USA, 2014, pp. 655–665.

[21] J. Pennington, R. Socher, C. D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 2014, pp. 1532–1543.

[22] D. Yogatama, M. Faruqui, C. Dyer, N. A. Smith, Learning word representations with hierarchical sparse coding, in: International Conference on Machine Learning, Lille, France, 2015, pp. 87–96.

[23] M. Faruqui, C. Dyer, Non-distributional word vector representations, Annual Meeting of the Association for Computational Linguistics, Beijing, China, 2015, pp. 464–469.

[24] P. Qian, X. Qiu, X. Huang, Investigating language universal and specific properties in word embeddings, in: Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1478–1488.

[25] J. Tang, M. Qu, M. Wang, et al., Line: Large-scale information network embedding, in: International Conference on World Wide Web, Florence, Italy, 2015, pp. 1067–1077.

[26] Q. V. Le, T. Mikolov, Distributed representations of sentences and documents, in: International Conference on Machine Learning, Beijing, China, 2014, pp. 1188–1196.

[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, E. Hovy, Hierarchical attention networks for document classification, in: Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, California, USA, 2016, pp. 1480–1489.

[28] W. Gao, N. Yoshinaga, N. Kaji, et al., Modeling user leniency and product popularity for sentiment classification, in: International Joint Conference on Natural Language Processing, Nagoya, Japan, 2013, pp. 1107–1111.

[29] L. Dong, F. Wei, M. Zhou, et al., Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis, in: AAAI Conference on Artificial Intelligence, Quebec City, Canada, 2014, pp. 1537–1543.

[30] D. Hovy, Demographic factors improve classification performance, in: Annual Meeting of the Association for Computational Linguistics, Beijing, China, 2015, pp. 752–762.

[31] C. Tan, L. Lee, J. Tang, et al., User-level sentiment analysis incorporating social networks, in: ACM Knowledge Discovery and Data Mining, San Diego, California, USA, 2011, pp. 1397–1405.

[32] X. Hu, L. Tang, J. Tang, et al., Exploiting social relations for sentiment analysis in microblogging, in: ACM International Conference on Web Search and Data Mining, Rome, Italy, 2013, pp. 537–546.

[33] D. Tang, B. Qin, T. Liu, Learning semantic representations of users and products for document level sentiment classification, in: Annual Meeting of the Association for Computational Linguistics, Beijing, China, 2015, pp. 1014–1023.

[34] Q. Diao, M. Qiu, C.-Y. Wu, et al., Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars), in: ACM Knowledge Discovery and Data Mining, New York, USA, 2014, pp. 193–202.

[35] S. Wang, C. D. Manning, Baselines and bigrams: Simple, good sentiment and topic classification, in: Annual Meeting of the Association for Computational Linguistics, Jeju Island, Korea, 2012, pp. 90–94.

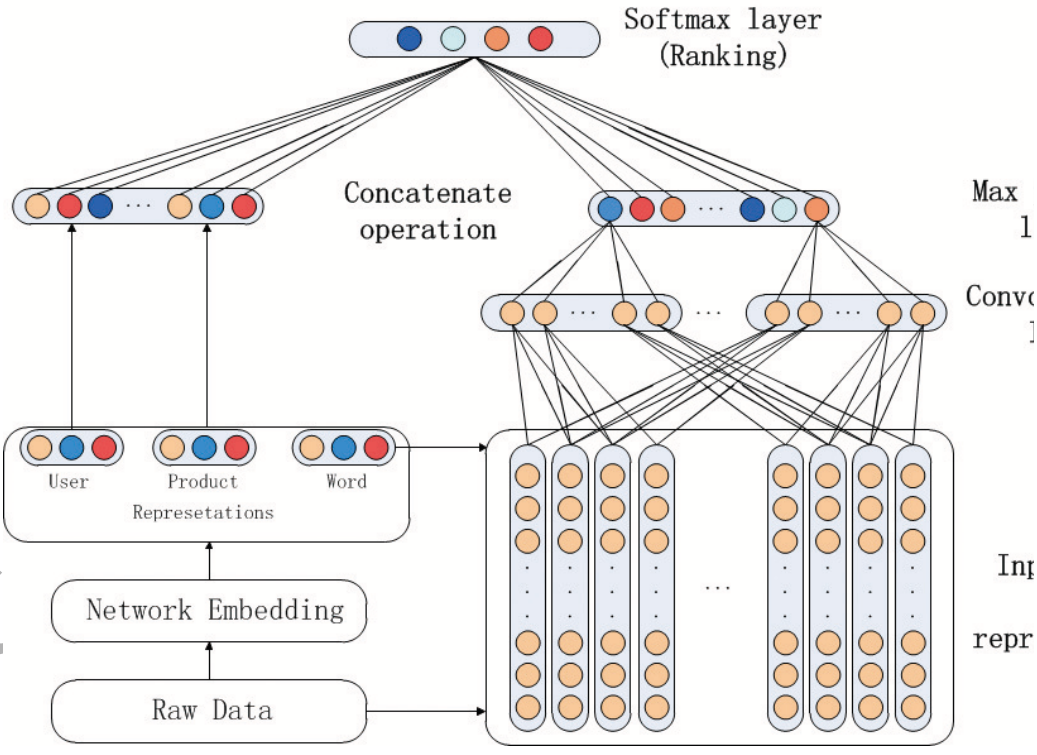
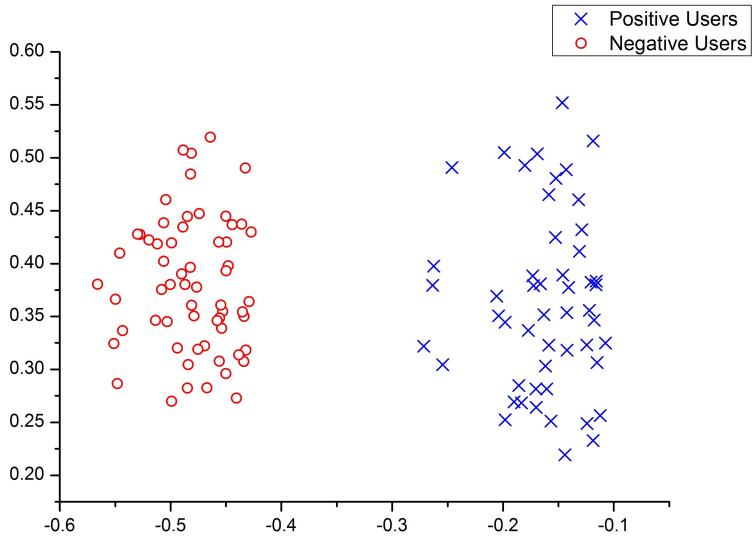
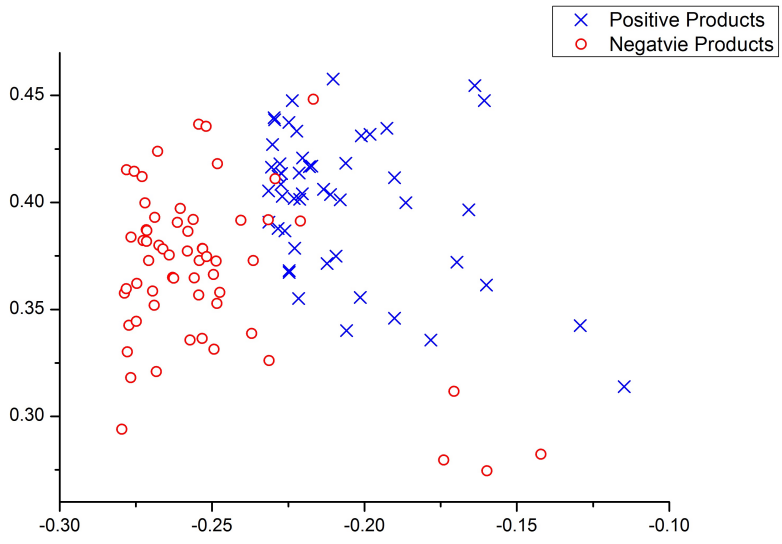


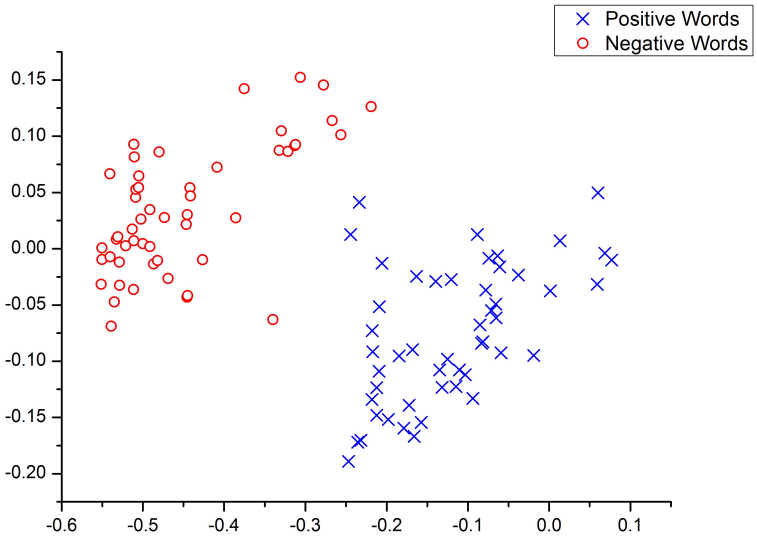
Figure 2: Architecture of heterogeneous embedded CNN.



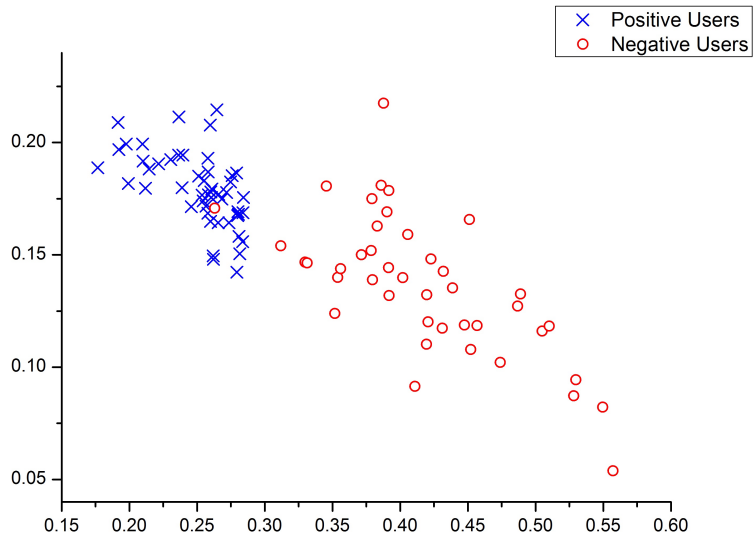
(a) Users in IMDB.



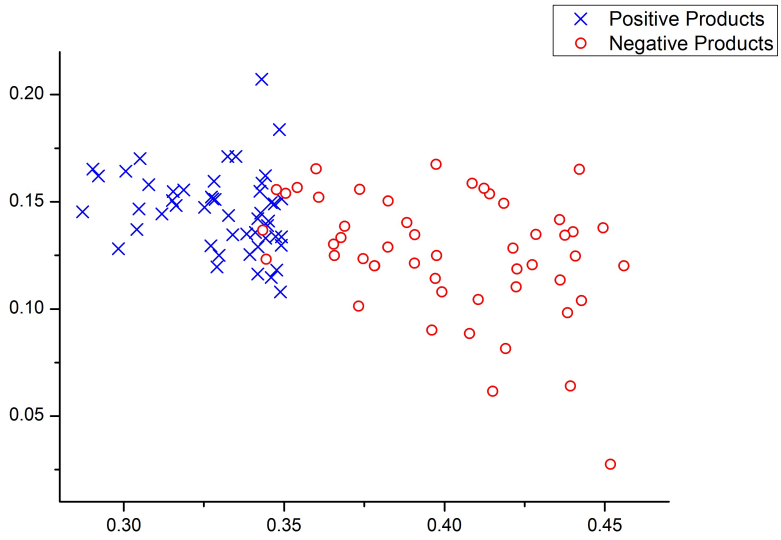
(b) Movies in IMDB.



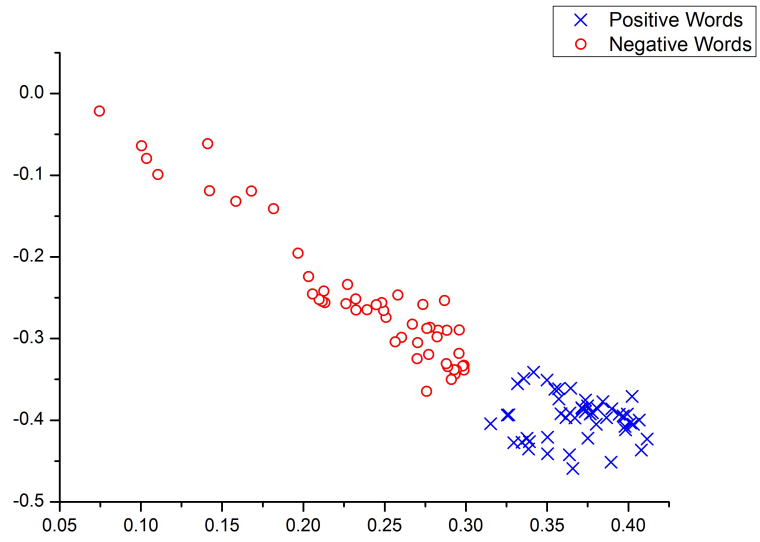
(c) Words in IMDB.



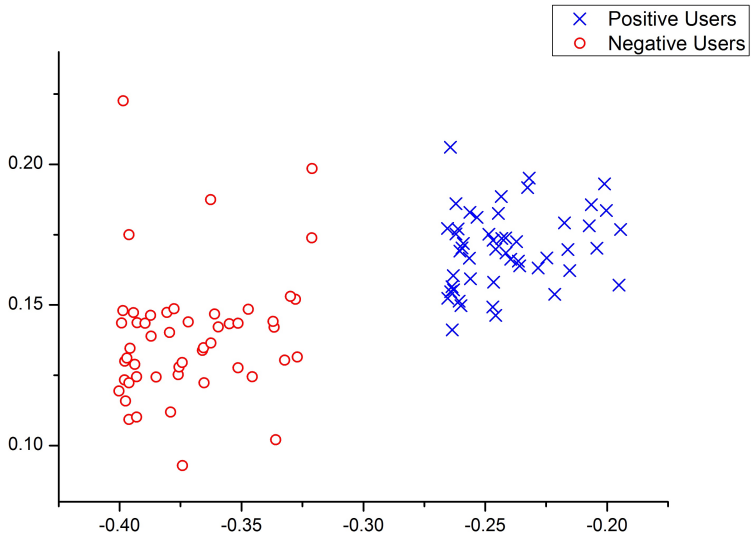
(d) Users in IMDB.



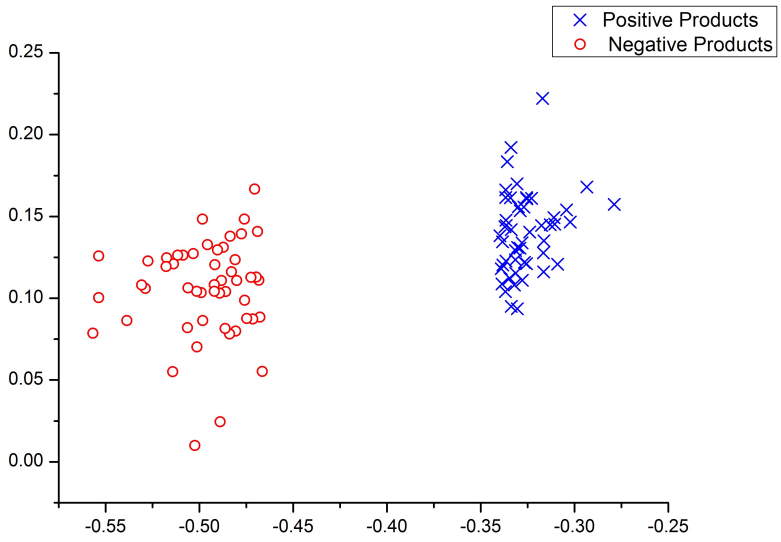
(e) Restaurants in IMDB.



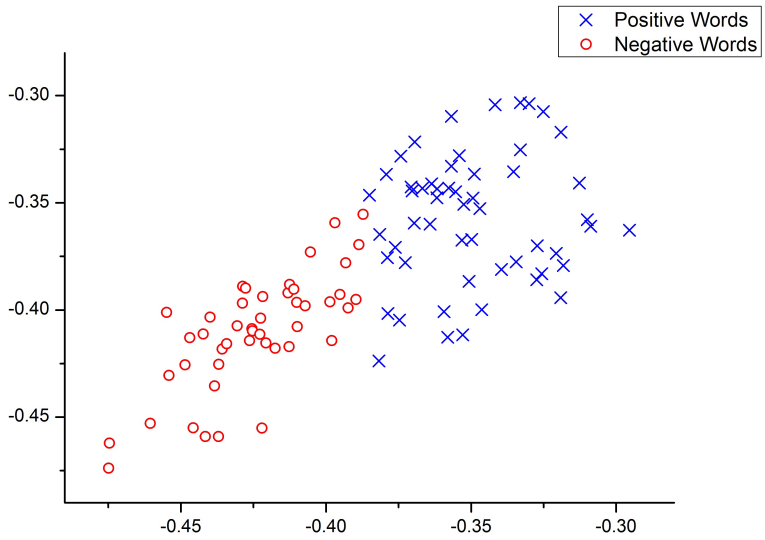
(f) Words in IMDB.



(g) Users in IMDB.



(h) Restaurants in IMDB.



(i) Words in IMDB.

Figure 3: The distribution of users, products and words in the embedding space (2D-PCA). Blue color denotes positive polarity while red color denotes negative polarity.