Task-specific Word Identification from Short Texts Using a Convolutional Neural Network

Shuhan Yuan *[1], Xintao Wu[2] and Yang Xiang[1]

[1]Computer Science and Technology Department,
Tongji University, Shanghai, China
Email:{4e66,shxiangyang}@tongji.edu.cn
[2]Computer Science and Computer Engineering Department,
University of Arkansas, Fayetteville, AR, USA
Email:xintaowu@uark.edu

Abstract

Task-specific word identification aims to choose the task-related words that best describe a short text. Existing approaches require well-defined seed words or lexical dictionaries (e.g., WordNet), which are often unavailable for many applications such as social discrimination detection and fake review detection. However, we often have a set of labeled short texts where each short text has a task-related class label, e.g., discriminatory or non-discriminatory, specified by users or learned by classification algorithms. In this paper, we focus on identifying task-specific words and phrases from short texts by exploiting their class labels rather than using seed words or lexical dictionaries. We consider the task-specific word and phrase identification as feature learning. We train a convolutional neural network over a set of labeled texts and use score vectors to localize the task-specific words and phrases. Experimental results on sentiment word identification show that our approach significantly outperforms existing methods. We further conduct two case studies to show the effectiveness of our approach. One case study on a crawled tweets dataset demonstrates that our approach can successfully capture the discrimination-related words/phrases. The other case study on fake review detection shows that our approach can identify the fake-review words/phrases.

Keywords: Task-specific word identification, convolutional neural network, deep learning

*Corresponding author: Shuhan Yuan, Computer Science and Technology Department, Tongji University, Shanghai, China. E-mail:4e66@tongji.edu.cn. This research was conducted while Shuhan Yuan visited University of Arkansas.

1 Introduction

Identifying task-specific words from a short text (e.g., tweet) aims to select the task-related words that best describe the short text. Task-specific word identification has received much attention in sentiment analysis research [10, 25, 13]. The sentiment words, which express a positive or negative polarity, are key information for text summarization and text filtering. However, these approaches often need to import either seed words or lexical dictionaries (e.g., WordNet) to identify the polarity words [10, 13].

In many applications (e.g., examining whether a tweet is discriminatory or contains private information), we do not have well-defined seed words or lexical dictionaries. Instead, we often have a set of labeled short texts each of which has a task-related class label, e.g., discriminatory/non-discriminatory or private/non-private, specified by users or learned by classification algorithms. In this work, we focus on identifying task-specific words and phrases from short texts by exploiting their class labels rather than using seed words or lexical dictionaries. For example, given a tweet "*Do you need any more proof the blacks are just low life Savages?*" and its label "*discriminatory*", we aim to identify the discrimination-related words such as "*blacks*" and "*savages*" from this tweet text.

Task-specific word identification is important for other text analysis tasks in natural language processing and information retrieval, such as text summary and lexical dictionary construction. To understand why a short text is related to its class, it is imperative to identify and highlight its task-specific words. The identified words can be considered as the discriminative features to separate the short text from ones in other classes. Moreover, the identified task-specific words from a large text corpus can be used to build dictionaries for many important and challenging tasks on social media analysis such as detecting racism-related discrimination tweets.

In this paper, we consider the task-specific word identification as feature learning [2] and use convolutional neural networks (CNNs) [21] to identify task-specific words from a set of labeled short texts. CNNs have achieved impressive performance in different feature learning problems in computer vision, speech recognition, and text analysis areas [9, 1, 20, 19, 17, 48]. Using CNNs to localize the positions of objects in an image based on the image-level labels has also been investigated [31, 53]. Our CNN-based approach for task-specific word identification is similar in principle to the image object localization task because both aim to find the most discriminative features from an input. However, the image object localization methods could not be directly adapted to task-specific word identification due to two reasons. First, the image object localization uses the raw data of an image as input to the CNN model whereas we cannot simply represent text as matrix. Second, the image object localization task is to highlight sub-regions of the input matrix, which correspond to physical objects in an image. So even if we derive the matrix representation of text, sub-regions identified by the object localization do not correspond to words or phrases.

To address the first issue, the words in a short text need to be mapped to their

feature space in our task. Traditionally, each word is represented as an one-hot vector which does not capture the semantic information about the word. Recently, mapping words to a low dimensional semantic vector space (called word embedding) is widely used as the representation of words [7, 28, 36]. We adopt word embeddings to represent the words in our task. Thus, we can construct a short text matrix by combining the word embeddings. For the second problem, simply selecting sub-dimensions of the text matrix is meaningless because sub-dimensions of word embeddings are generally unexplainable. We propose an approach to determine a hidden score vector that can quantify the importance or relevancy of words to a specific task. The words with high score values in the score vector are then considered as task-specific words.

Our task-specific word identification approach is based on training a CNN over a set of labeled texts. We show how to derive the score vectors using the parameters of the CNN model built from the set of labeled texts. We further extend our task-specific word identification approach to task-specific phrase identification. We conduct three experiments to evaluate our approach, sentiment word identification, discrimination-related word/phrase identification, and fake review word/phrase identification. The first experiment shows that our approach can more accurately identify sentiment words than existing approaches by comparing the identified words with the ground truth. The second experiment demonstrates our approach can successfully capture meaningful discrimination-related words/phrases from a crawled tweet dataset. The third experiment shows our approach can figure out the strong polarity words which are frequently in fake reviews.

The rest of this paper is organized as follows. In section 2, we first briefly review of the related work on feature selection and sentiment word identification, along with research on deep neural networks for short text modeling. We then introduce our model for task-specific word identification, which computes a score vector for evaluating the weights of words in a short text by using a well-trained convolutional neural network. To evaluate our model, we conduct three experiments, which are sentiment word identification, discrimination-related word/phrase identification and fake review word/phrase identification. The experimental results show the effectiveness of our model. Finally, we conclude our work.

2 Related Work

2.1 Feature selection and sentiment word identification

The feature selection methods for text classification can adapt to identify the task-specific words. The feature selection can reduce the dimension of input space and identify the discriminative features to improve the performance of classification. There are a large number of unsupervised feature selection methods based on statistical measures, like term frequency, information gain, mutual information, and term strength [8, 47, 27, 44]. Some other methods are based

on dimensionality reduction including principle component analysis (PCA), linear discriminant analysis, and locally linear embedding [24]. However, feature selection as a preprocessing step of data mining and machine learning applications is used for classification or clustering. Our method combines the label information of the text to identify the discriminative features, which can further improve the performance of selecting task-specific words.

Sentiment word identification, as a special case of task-specific word identification, usually requires seed words. Some of the approaches are based on the similarity between the words and seed words. For example, [43, 37] adopt point-wise mutual information to measure the similarity. [18] assumes the same polarities would appear successively of the seed words in contexts and defines the context coherent to measure the similarity. [16] aims to generate topic-specific subjectivity lexicons from a general polarity lexicon using a bootstrapping method. However, the seed words are selected manually and domain-dependence, which are usually incomplete. Heavy relying on seed words restricts the performance of identifying the task-specific words. Recently, some other methods [49, 25, 35] are proposed to identify sentiment words based on optimization models without using seed words. [26] focuses on identifying the sentiment lexicons which have different meanings in different aspects using an optimization framework. However, these methods are limited to identify sentiment words from documents and are not suitable for short texts. Meanwhile, topic models such as Latent Dirichlet allocation (LDA) [4] can identify topic words from a corpus as task-specific words. [52] further incorporate appraisal expression patterns to LDA for aspect and sentiment word identification. However, LDA usually requires a large number of documents and is not suitable for short texts either [42].

Closely related to our approach is weakly supervised class saliency maps [38, 22] which are widely used in computer vision area for object localization and class saliency visualization. However, as shown in our experiment evaluation, the performance of class saliency maps on task-specific words is poor. In this work, we propose a weakly supervised method to identify task-specific words from short texts based on the convolutional neural network. Our method is trained on labeled text corpus, so no seed words or other additional annotation are required. Meanwhile, our method further combines the label information to identify the task-specific words. Thus, our method is suitable for applications such as social discrimination detection from texts and fake review detection.

2.2 Deep neural networks for short text modeling

Deep neural networks have achieved promising results in natural language processing, like text classification [19], question answering [45, 5], and machine reading [14, 6].

The fundamental of applying the deep neural networks for natural language processing is word embeddings [28, 3] which map each word to a dense vector. These word embeddings are trained in an unsupervised way on a large text corpus. Word embeddings can avoid the use of hand-designed features and capture

the hidden semantic and grammatical features of words. Thus, word embeddings can improve the performance of many natural language processing tasks [7]. To further compose the representations of phrases and short texts, the idea of semantic composition is applied to the word embeddings. The basic model is based on the algebraic operations, like additive and multiplication, to build the short text vector from word embeddings [29]. However, simple algebraic operations cannot capture the complicated structure of the natural language. Some complex models based on deep neural networks are proposed recent years. The recursive neural network [40] can construct the grammar tree-like structure to represent the short text in order to capture the grammar information. The recurrent neural network [12] processes the short text word by word in order, which capture the sequential information of a short text. Researchers also proposed a tree-structured recurrent neural network [41] to combine the advantages of recursive neural network and recurrent neural network.

In this work, we adopt the convolutional neural network which as a primary model of deep neural networks has also achieved great performance on different areas, like computer vision [20], speech recognition [1] and natural language processing [19]. Because of somewhat opaque of CNN, researchers try to demystify and understand why the performance of CNN is promising. In computer vision area, saliency maps [38, 22] and deconvolution networks [51, 50] are proposed to explain the model. However, there is little work to explain the internal operation and behavior of CNN model working on the text. Our work can provide some insights about the CNN on text classification.

3 Model Description

In this section, we describe our approach for identifying task-specific words or phrases. Our approach first trains the CNN model and transfers the well-learned representation of sentence to identify the task-specific words and phrases. We first introduce how to construct score vectors to identify the task-specific words based on the convolutional neural network. Then, we extend our approach to identify the task-specific phrases.

3.1 Problem Statement

Given a corpus of labeled short texts $\mathcal{X}$, each short text contains n words and has the class label c, which can be described as $X = [x_1, x_2, \ldots, x_n]$ where x_i denotes the i-th word and the class label c is from a set C. We assume that there is a hidden score vector $\mathbf{s}_c = [s_1, s_2, \ldots, s_n]$ for the short text X where s_i measures the importance or relevancy of the word x_i to the class c. Similarly, there is a score vector to measure the importance of corresponding phrases for class c. Thus, our task is to derive the score vector $\mathbf{s}_c$ based on the text X and its label c to locate task specific words or phrases. Important notations are summarized in Table 1.

3.2 Convolutional Neural Network for short text Representation

The short text representation by deep learning models first uses the word embeddings to represent the words in the short text and then performs a semantic composition over the word embeddings to build the short text representation. In this work, our task-specific word identification approach is based on the convolutional neural network first introduced in [19] to compose the short text representation. In this section, we first give a brief review about the CNN model for short text representation.

Using CNN to model the short text representation, we first map each word x_i in the text X to a d-dimensional real-valued vector space $\mathbf{x}_i \in \mathbb{R}^d$. These word embeddings are trained in an unsupervised way on a large text corpus. Word embeddings can avoid the use of hand-designed features and capture the hidden semantic and grammatical features of words. An embedding matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$ is then constructed by combining each word embedding of the text, where $\mathbf{X} \in \mathbb{R}^{n*d}$.

Then, a convolution operation which involves a *filter* $\mathbf{W} \in \mathbb{R}^{h*d}$ is applied to h continuous word embeddings $\mathbf{X}_{j:j+h-1}$ to generate a hidden feature of these h words:

$$v_j = g(\mathbf{W} * \mathbf{x}_{j:j+h-1} + b), \tag{1}$$

where b is the bias; $*$ is a two-dimensional convolution operation and g indicates a non-linear function. The filter slides through the whole short text matrix $\mathbf{X}$ and generates a feature vector $\mathbf{v} = [v_1, v_2, \ldots, v_{n-h+1}]$. After that, a pooling operation is applied to the feature vector $\mathbf{v}$. There are two widely used pooling operation: max pooling and average pooling. The max pooling operation aims to capture the most superior part of the feature vector by keeping its highest value and is defined as:

$$r_{max} = max(v_1, v_2, ..., v_{n-h+1}). \tag{2}$$

The average pooling aims to capture all discriminative features of a text and is defined as:

$$r_{avg} = \frac{1}{n-h+1} \sum_{j=1}^{n-h+1} v_j. \tag{3}$$

To capture different aspects of features from the text, the model usually applies multiple filters with different sizes of windows h to generate the feature vector. After applying the pooling operation on each feature vector, the model encodes the input short text to a representation vector $\mathbf{r} = [r_1, r_2, \cdots, r_m]$, where m is the number of feature vectors generated by m different filters. A softmax classifier is applied on top of the representation vector $\mathbf{r}$ to predict the labels of the short texts. The softmax function is defined as:

$$p(y = c|\mathbf{r}; \mathbf{U}) = \frac{exp(\mathbf{u}_c^T \mathbf{r})}{\sum_{c' \in C} exp(\mathbf{u}_{c'}^T \mathbf{r})}, \tag{4}$$

where $\mathbf{U}$ is the parameters of the softmax function and $\mathbf{u}_c$ is the c-th column of $\mathbf{U}$; and c is the class of the given text. We adopt the cross-entropy loss function to train the CNN model:

$$L(y) = -\log P(y = c|\mathbf{r}; \mathbf{U}). \tag{5}$$

The parameters of the model are updated by the backpropagation algorithm.

Algorithm 1: Task-specific Word/Phrase Identification

Inputs : Training dataset $\mathcal{X}$, maximum training epoch *Epoch*, number of filters m, filter size h

Outputs: Task-specific words for each text X in $\mathcal{X}$

```
k ← 0;
while k < Epoch do
    for each short text X in 𝒳 do
        Compose the text matrix X for X using word embeddings;
        l ← 0;
        for l < m do
            Generate feature vector v by applying the filter W ∈ ℝ^{h*d} on
             the text matrix X based on Eq. 1;
            Apply the pooling operation on v by Eq. 2 for max pooling or
             Eq. 3 for average pooling to get hidden feature value r;
            l ← l + 1;
        end
        Construct the short text representation vector r = [r_1, r_2, ··· , r_m];
        Compute the softmax function p(y = c|r; U) by Eq. 11;
        Compute the loss function L(y) by Eq. 5;
        Update parameters of CNN by the backpropagation algorithm;
    end
    k ← k + 1;
end
for each short text X in 𝒳 do
    Use the trained CNN model to generate feature vectors of text X by
     Eq. 7;
    Generate score vector s_c by Eq. 12;
    Get task-specific words for text X which have the Top-k highest
     values in the score vector s_c
end
return the task-specific words for each text X in 𝒳
```

3.3 Task-specific Word Identification

To identify the task-specific words, we adopt a CNN model to learn the score vector $\mathbf{s}_c$ based on the text $X = [x_1, x_2, \ldots, x_n]$ and its label c. The text words

having the Top-k highest values in the score vector $\mathbf{s}_c$ are then highlighted as task-specific words.

Given a short text $X = [x_1, x_2, \ldots, x_n]$ with n words, we first map the short text to a short text matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, where $\mathbf{X} \in \mathbb{R}^{n*d}$ and $\mathbf{x}_n$ is the corresponding word embedding of the word x_n. We apply the one word *filter* $\mathbf{w} \in \mathbb{R}^d$ to each word embedding in the matrix $\mathbf{X}$ to build a convolutional layer. For word x_j, we learn its feature v_j by:

$$v_j = g(\mathbf{w} * \mathbf{x}_j + b), \tag{6}$$

where b is the bias; $*$ indicates the one-dimensional convolution operation and g is a non-linear function. The feature v_j is trained to represent a linguistic feature of the word x_j in X. Then, the filter passes through the whole matrix $\mathbf{X}$ and produces a feature vector:

$$\mathbf{v} = [v_1, v_2, \ldots, v_n]. \tag{7}$$

After obtaining the feature vector $\mathbf{v}$, a pooling operation is applied. The pooling operation is able to capture the salient information of the feature vector $\mathbf{v}$. In our model, we adopt both the max pooling and average pooling operation to extract the feature from the feature vector as the feature extracting from one filter. The max pooling operation extracts the maximum value $r_{max} = max(\mathbf{v})$ of the feature vector $\mathbf{v}$:

$$r_{max} = max(v_1, v_2, ..., v_n). \tag{8}$$

The average pooling operation computes the mean value $r_{avg} = mean(\mathbf{v})$ of the feature vector $\mathbf{v}$:

$$r_{avg} = \frac{1}{n} \sum_{j=1}^{n} v_j. \tag{9}$$

Note that one feature vector $\mathbf{v}$ is computed by the same filter $\mathbf{w}$ in a CNN. Thus, the feature vector $\mathbf{v}$ actually captures one hidden semantic meaning of the input text. For modeling the short text, we need more feature vectors to capture the different aspects of hidden semantic meanings. In our work, we derive m feature vectors, $\mathbf{v}_1, \cdots, \mathbf{v}_m$. Each feature vector $\mathbf{v}_i$ is computed by a unique $\mathbf{w}_i$ and b_i. After applying the pooling operation on each feature vector $\mathbf{v}_i$, we get its corresponding feature r_i and then combine them in one vector $\mathbf{r}$:

$$\mathbf{r} = [r_1, r_2, \cdots, r_m]. \tag{10}$$

In order to learn the representation vector $\mathbf{r}$ of X, we train the CNN model in a supervised way to predict the class of text X by using a softmax function:

$$p(y = c|\mathbf{r}; \mathbf{U}) = \frac{exp(\mathbf{u}_c^T \mathbf{r})}{\sum_{c' \in C} exp(\mathbf{u}_{c'}^T \mathbf{r})}, \tag{11}$$

where $\mathbf{U}$ is the parameters of the softmax function and $\mathbf{u}_c$ is the c-th column of $\mathbf{U}$; and c is the class of the given text. We adopt the cross entropy loss

function defined in Equation 5 to train the model. After training the CNN for classification, the representation vector $\mathbf{r}$ expects to encode the semantic information about the input text and its class.

Next we show how we derive the score vector $\mathbf{s}_c = [s_1, s_2, \ldots, s_n]$ by using parameters $\mathbf{U}$ and feature vectors $\mathbf{v}_1, \cdots, \mathbf{v}_m$, all of which are well trained in the CNN. From the softmax function shown in Equation 11, we can notice that $\mathbf{u}_c^T \mathbf{r} = \sum_i u_{ic} r_i$, where u_{ic} is the entry in the i-th row and the c-th column of softmax parameter $\mathbf{U}$. Therefore, we can consider u_{ic} as the weight corresponding to feature r_i for class c. This also indicates that u_{ic} captures the importance of each feature vector $\mathbf{v}_i$ for predicting class c, because r_i is computed by the pooling operation on $\mathbf{v}_i$. Hence, the score vector $\mathbf{s}_c$ is computed by projecting the weights of softmax on to feature vectors $\mathbf{v}_i$:

$$\mathbf{s}_c = \sum_{i=1}^{m} u_{ic} \mathbf{v}_i. \tag{12}$$

We illustrate the procedure of generating score vectors using an example in Figure 1. Recall that each value in $\mathbf{v}_i$ measures one aspect importance of a word to a text X, and its weight u_{ic} measures the importance to the class c. Thus, based on Equation 12, the values in $\mathbf{s}_c$ indicate the importance scores of words in a given text for class c. For example, if the i-th value is the highest value in the score vector $\mathbf{s}_c$, the corresponding word x_i contains the most important information about class c. Then, we select Top-k words which have the highest score values in score vector $\mathbf{s}_c$ as the task-specific words. Our approach also provides a way to understand the internal behavior of the CNN model. The words with the corresponding higher values in the score vector $\mathbf{s}_c$ are the key information for the CNN model to predict the label of a sentence.

We show pseudo code of our approach in Algorithm 1. For the task-specific word identification, we set the filter size $h = 1$. We train the CNN model in Lines 2-17 and use the trained CNN model to identify task-specific words in Lines 18-22. For each text X in the training data $\mathcal{X}$, we first compose its text matrix $\mathbf{X}$ using word embeddings in Line 4. We generate feature vector $\mathbf{v}$ by applying the *filter* $\mathbf{W} \in \mathbb{R}^{h*d}$ on the text matrix $\mathbf{X}$ based on Eq. 1 in Line 7. We then get hidden feature value r by applying the pooling operation on $\mathbf{v}$ by Eq. 2 for max pooling or Eq. 3 for average pooling in Line 8. After applying multiple filters, we construct the short text representation vector $\mathbf{r} = [r_1, r_2, \cdots, r_m]$ in Line 11 and predict text label by using the softmax function $p(y = c|\mathbf{r}; \mathbf{U})$ by Eq. 11 in Line 12. Finally we compute the loss function $L(y)$ by Eq. 5 in Line 13 and update parameters of CNN by the backpropagation algorithm in Line 14. We then use the trained CNN model to generate feature vectors of text X by Eq. 7 in Line 19. We apply Eq. 12 to generate score vector $\mathbf{s}_c$ in Line 20 and derive task-specific words for text X which have the Top-k highest values in the score vector $\mathbf{s}_c$ in Line 21.

3.4 Task-specific Phrase Identification

A phrase is defined as a concatenation of h continuous words (h-grams). For the text $X = [x_1, x_2, \ldots, x_n]$, we have its phrases $P_X = [p_1, p_2, \cdots, p_{n-h+1}]$, where p_j denotes the concatenation of words $x_j, x_{j+1}, \cdots, x_{j+h-1}$. Thus, the j-th phrase embedding is $\mathbf{X}_{j:j+h-1} = [\mathbf{x}_j, \mathbf{x}_{j+1}, \cdots, \mathbf{x}_{j+h-1}]$. To identify task-specific phrases from each text X, we adopt the same idea of deriving a score vector $\mathbf{s}_c = [s_1, s_2, \ldots, s_{n-h+1}]$ to measure the importance of each phrase in the text.

We set a phrase filter $\mathbf{W} \in \mathbb{R}^{h*d}$ to cover a window of h words. Thus, in the CNN model, each feature v_j captures the hidden feature of phrase p_j about class c by applying the filter $\mathbf{W}$ to $\mathbf{X}$:

$$v_j = g(\mathbf{W} * \mathbf{X}_{j:j+h-1} + b), \tag{13}$$

where b is the bias; $*$ is the two-dimensional convolution operation and g indicates a non-linear function. Then, the feature vector is $\mathbf{v} = [v_1, v_2, \ldots, v_{n-h+1}]$. The max pooling operation is defined as:

$$r_{max} = max(v_1, v_2, ..., v_{n-h+1}). \tag{14}$$

The average pooling is defined as:

$$r_{avg} = \frac{1}{n-h+1} \sum_{j=1}^{n-h+1} v_j. \tag{15}$$

We follow the same procedure described in Algorithm 1 with the filter size h to first train the CNN model for text classification, produce the score vector $\mathbf{s}_c$, and output task-specific phrases with length h.

4 Experiments

To evaluate the effectiveness of our approach, we conduct three experiments. In the first experiment, we focus on identifying sentiment words using two publicly available datasets with the ground truth. In the second experiment, we conduct a case study of identifying social discrimination-related words or phrases from tweets. We crawl our own datasets and focus on two types of social discriminations, sexism and racism. In the third experiment, we conduct another case study of identifying fake review words or phrases based on a fake review dataset.

Word Embeddings and Hyperparameters. We use the off-the-shelf pre-trained word embeddings (`https://code.google.com/archive/p/word2vec/`) [28] and randomly initialize the words that do not have pre-trained word embeddings. The dimension of word embeddings d is 300. The number of feature vectors m is 100.

Baselines. In our approach, we evaluate two pooling operations in our CNN model and name the score vectors with average pooling operation and max pooling operation **SV-AVG** and **SV-MAX**, respectively. We compare our approach with the following baselines for task-specific word identification.

- TF-IDF: TF-IDF, widely used as a feature selection method in information retrieval and text mining, is a statistic to reflect how important a word is to a document in a collection or corpus. We calculate the TF-IDF value of words on the positive and negative text corpus separately. For each text, we output the Top-k words with the highest TF-IDF values.

- TF-IDF-softmax: We apply the softmax classifier on the TF-IDF features. After training the model for text classification, we follow the similar procedure described in last section to identify the task-specific words. The score vector is generated by multiplying the parameters $\mathbf{u}_c$ of softmax with TF-IDF values of each text for the class c.

- Saliency Map: Saliency Map was proposed as a CNN visualization technique [38]. It computes the gradient of the class score with respect to the input and can identify the discriminative features of the input. The authors [11] applied this technique to highlight the important sentences of a document. In our experiment, we consider the words in a text with high absolute values of the derivative on word embeddings as task-specific words. The saliency map can be derived based on CNN with max pooling or CNN with average pooling. We denote the saliency map using CNN with max pooling (average pooling) as **SalMap-MAX** (**SalMap-AVG**).

Evaluation Metric. For sentiment word and discrimination-related word identification, we adopt accuracy@k, precision, recall, and F1 to evaluate the performance.

- *Accuracy@k* is a metric which calculates the fraction of the Top-k words selected by each method with the ground truth T.

- We further evaluate our approach based on the *precision*,*recall* and *F1*. We obtain the Top-k words of each text with highest values in its score vector and compare the selected words with the ground truth dictionary.

4.1 Sentiment Word Identification

The first experiment is to identify the sentiment words from each text. We only consider the binary classification problem, so the objective of our model is to select the positive and negative sentiment words.

Datasets. We evaluate the performance of sentiment word identification on two datasets — the Movie Review dataset (**MR**) [34] and the Stanford Sentiment Treebank dataset (**SST**) [39]. The average length of texts in MR is 20 and that of SST is 19. We remove the neutral reviews and binarize labels for SST. We compose the ground truth dictionary T by combing the MPQA subjective lexicon dataset [46] and the sentiment lexicon dataset [15]. Only the strong subjective words in MPQA are considered as sentiment words. The ground truth dictionary T contains 2006 positive words and 4783 negative words.

Results. We use 10-fold cross-validation to evaluate the performance of all methods except the TF-IDF. In each fold, we use the training dataset to train

the models for sentiment classification and use the test dataset for sentiment word identification. Note that TF-IDF is a count-based feature selection method and the TF-IDF values are fixed for a given dataset. Thus, we do not need to conduct cross validation for TF-IDF. We select the Top-k words with the highest TF-IDF values and compare with the ground truth.

Tables 2 and 3 show experimental comparisons of our methods, SV-AVG and SV-MAX, and three baselines in terms of the accuracy@k, precision, recall and F1 metrics. Tables 2 first presents the sentiment classification accuracy. Results of the Accuracy@k are then shown in columns "Top-k". Note that the TF-IDF and Saliency Map methods are feature selection methods which do not predict the class labels. Thus, we can not report the sentiment classification results for these two methods. We have the following observations:

(1). The SV-AVG method achieves the best performance on sentiment word identification, although its classification accuracy is slightly worse than the SV-MAX. Recall that the max pooling only keeps the highest value whereas the average pooling combines all values. Thus, the max pooling operation is good at finding the most discriminative features to separate texts into different classes because the classifier can predict the class accurately with the most discriminative features. On the contrary, the performance of SV-AVG for classification is not as good as SV-MAX because SV-AVG decreases the values of the most discriminative features by averaging all the feature values. In our sentiment word identification scenario, SV-MAX drops too much information by only keeping the maximum value, but SV-AVG with averaging the feature values keeps more useful information in the neural network. Therefore, SV-AVG is suitable for identifying all the discriminative words.

(2). The TF-IDF-softmax method performs better than the TF-IDF method, which indicates that supervised training the classifier can encode task-specific information in its parameters. Although the TF-IDF-softmax adopts the same classifier as our models, the performance of TF-IDF-softmax is worse than our methods. This demonstrates that applying the convolutional operation on word embedding can capture more semantic information than the statistical features of TF-IDF.

(3). Although Saliency Map (SalMap-Max, SalMap-AVG) adopts the same CNN model, the performance of Saliency Map is only slightly better than the TF-IDF method and worse than our SV-AVG and SV-MAX. This indicates that the Saliency Map on word embedding captures much less information about identifying task-related words than the softmax parameters. We can further observe that the performance of SalMap-AVG is worse than SalMap-MAX. This is because CNN with the max pooling operation can achieve better classification accuracy than using the mean pooling and the computed saliency map has larger gradient values to the discriminative features which are useful for identifying the task-specific words.

(4). We also notice that all methods generally achieve higher accuracy in SST than MR. This is because more texts in SST contain sentiment words than MR. It also indicates a positive correlation between sentiment word identification and classification accuracy.

We further compare the sentiment words selected by our models and baselines. Table 4 shows the Top-10 most frequent words selected by each methods. The words are listed in order based on their frequency values in Top-5 lists. Comparing with the baselines, the Top-10 words selected by SV-AVG and SV-MAX are almost sentiment words; and the sentiment words are in the correct categories. It indicates the CNN model can figure out the most discriminative features from a sentence automatically. That is the reason why the CNN model achieves better performance for text classification than other methods, especially those methods which use the same classifier (i.e., TF-IDF-softmax).

4.2 Social Discrimination-related Word and Phrase Identification

In this experiment, we aim to identify discrimination-related words and phrases (sexism-related words/ phrases or racism-related words/phrases) from tweets. These tweets are automatically labeled as about *sexism*-related or *racism*-related by hashtags. For example, if a tweet contains hashtag "#sexism", we consider this tweet is related with sexism.

Datasets. We crawled tweets during the period from November 1, 2015 to April 17, 2016, which contain *sexism* or *racism* hashtags. We pre-process the tweets by tokenizing them and removing all punctuation and tokens beginning with the "@" symbol. We keep the tweets that contain more than 10 words. For those hashtags in tweets, we remove the hash signs and keep the words.

To evaluate the accuracy, we create a dataset T_1 containing 300 well-labeled tweets. In each tweet, the discrimination-related words are marked by two domain experts. The dataset is composed by 150 sexism-related tweets and 150 racism-related words. Meanwhile, to further conduct a case analysis about discrimination-related word and phrase identification, we compose another dataset T_2 which contains 10000 tweets with hashtag "#sexism" and 10000 tweets with hashtag "#racism".

Results of discrimination-related word identification. To evaluate the precision, recall and F1, we train all methods except the TF-IDF with 5 fold cross-validation on dataset T_1. Table 5 shows the precision, recall and F1 of discrimination-related words identification. In this experiment, we adopt the CNN model with the max pooling operation to compute the saliency map. The overall results are similar to the results of sentiment word identification. Our methods achieve the best performance on identifying the discrimination-related words.

Then, we conduct a detailed analysis about each method for task-specific word identification on dataset T_2. For each method, we build a dictionary of 100 words that occur most frequently in the Top-5 list of a tweet. We compare the dictionaries by calculating the overlapping size between the dictionary created by SV-AVG and that by each baseline. We find significant differences. Specifically, the ratios of overlapping words selected by SV-AVG and TF-IDF, TF-IDF-softmax, Saliency Map are 36%, 58% and 36%, respectively.

Table 6 shows the Top-10 most frequent words selected by each methods. The words are listed in order based on their frequency values in Top-5 list. Comparing with the baselines, the Top-10 words selected by SV-AVG and SV-MAX contain more discrimination-related words. This is because SV-AVG and SV-MAX capture class information during the training process and filter out those words that are not related to the specific task.

Table 7 shows several tweet examples where discrimination-related words selected by SV-AVG are highlighted. A word highlighted with the red color means the word in the Top-5 list. The superscript of a word indicates its ranking based on the score vector. We can see that our method can locate the discrimination-related words with high scores. Furthermore, our method can identify the words which are closely related to task-specific words such as "girls", "#white", and "#Africa".

Results of discrimination-related phrase identification. For phrase identification, we set the phrase *filter* $h = 2$ and focus on identifying the discrimination-related phrases containing 2 words. Table 8 shows several tweet examples where the Top-3 discrimination-related phrases are highlighted based on the score vector. Some phrases are longer than 2 words because the stop words are removed during the training phase. Our model successfully identifies the discrimination-related phrases such as "black judge", "gender bias". Meanwhile, the identified discrimination-related phrases usually contain the discrimination-related words such as "sexism", "racism", "women" and "black".

4.3 Fake Review Word and Phrase Identification

In this experiment, we focus on identifying the words and phrases which are related to the fake review. The fake review detection aims to identify the reviews which try to mislead readers. There are a large number of websites (e.g., Amazon (`https://www.amazon.com`) and Yelp (`https://www.yelp.com`)) which allow people to review the products or services which the users purchased. Many people rely on reviews for buying anything online. However, many reviews are generated by paid reviewers or rivals, which make the reviews unreliable. The fake review detection becomes a hot research topic recent years [33, 30, 23]. In this work, we aim to figure out the fake review words and phrases, which can show the effectiveness of our model and further help to understand the pattern of fake reviews. Our model can also use to build blacklist words about the fake reviews, which can help to filter out the fake review automatically.

Datasets. We use the fake review corpus provided by [33, 32], in which the truthful positive reviews were crawled from TripAdvisor and the deceptive positive reviews were submitted by Amazon Mechanical Turk. The reviews are about the most popular Chicago hotels. Each category of the corpus contains 400 reviews. In our experiment, we focus on identifying the truthful positive review words and deceptive positive review words.

Results of fake review word and phrase identification. Because there aren't ground-truth fake review words and phrases, in this experiment, we conduct a case analysis about fake review word and phrase identification by showing

the words and phrases selected by our methods. We first show the Top-10 most frequent words from truthful and deceptive positive reviews selected by our method and baselines in Table 9. Due to the space limit, we only report the result of SV-MAX. The words are listed in order based on their frequency values in Top-5 list.

We first focus on the words selected by SV-MAX. We can see that comparing with the first two words "floor" and "bathroom" in Top-10 truthful positive review words, the first words "hotel" and "room" in Top-10 deceptive positive review words are more general words to describe a hotel. Meanwhile, the rest of words in deceptive positive reviews contain highly polarized adjectives. Thus, we can figure out that the fake reviews prefer to use strong adjectives to describe the hotels. In contrast, the truth reviews have more nouns and moderate words related to the hotels. It also indicates that comparing with the true reviews, the CNN classifier assigns high weights to the adjectives to identify the fake reviews. We further compare the words selected by SV-MAX and baselines. In general, SV-MAX identifies more strong polarized words from deceptive positive reviews than the baselines, which indicates the effectiveness of our method.

For fake review phrase identification, we set the *filter* $h = 2$ and compare the phrases between truthful positive reviews and deceptive positive reviews selected by SV-MAX. Table 10 shows one review example of each category where the Top-3 category-related phrases are highlighted based on the score vector. In general, our model can highlight the fake review phrases. Meanwhile, we can further see that although the truthful reviews also contain the sentiment words, the CNN classifier uses the nouns (i.e., locations) to identify the true reviews. It also means that the truthful reviews contain more specific description about the hotels.

5 Conclusion

In this paper, we have focused on task-specific word or phrase identification based on the convolutional neural network. We show how to derive the hidden score vector from CNN parameters and why the score vector can be used to identify task-specific words or phrases. Experimental results on sentiment word identification showed that our approach can significantly improve the accuracy for identifying the task-specific words compared with state-of-the-art methods including TF-IDF, softmax classifier and saliency map. We further showed that our approach can successfully identify the task-specific words or phrases effectively from discrimination-related tweets and fake review dataset. In our future work, we will extend our approach to identify key sentences from a document by using the convolutional document model.

Acknowledgements

The authors acknowledge the support from the National Natural Science Foundation of China (71571136), the 973 Program of China (2014CB340404), and the Research Projects of Science and Technology Commission of Shanghai Municipality (16JC1403000, 14511108002) to Shuhan Yuan and Yang Xiang, and from National Science Foundation (1646654,1564250) to Xintao Wu.

References

[1] Ossama Abdel-Hamid, Li Deng, and Dong Yu. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech*, 2013.

[2] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[3] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[5] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620, 2014.

[6] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, 2016.

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

[8] Anirban Dasgupta, Petros Drineas, Boulos Harb, Vanja Josifovski, and Michael W. Mahoney. Feature selection methods for text classification. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 230–239, 2007.

[9] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[10] Zhi-Hong Deng, Hongliang Yu, and Yunlun Yang. Identifying sentiment words using an optimization model with l1 regularization. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 115–121, 2016.

[11] Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. Modelling, visualising and summarising documents with a single convolutional neural network. *arXiv:1406.3830 [cs, stat]*, 2014.

[12] Alex Graves. Generating sequences with recurrent neural networks. *arXiv:1308.0850 [cs]*, 2013.

[13] Ahmed Hassan and Dragomir Radev. Identifying text polarity using random walks. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 395–403, 2010.

[14] Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, 2015.

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

[16] Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. Generating focused topic-specific sentiment lexicons. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 585–594, 2010.

[17] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

[18] Hiroshi Kanayama and Tetsuya Nasukawa. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363, 2006.

[19] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105, 2012.

[21] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[22] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. Visualizing and understanding neural models in nlp. *arXiv:1506.01066 [cs]*, 2015.

[23] Jiwei Li, Myle Ott, Claire Cardie, and Eduard Hovy. Towards a general rule for identifying deceptive opinion spam. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1566–1576, 2014.

[24] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *arXiv:1601.07996 [cs]*, 2016.

[25] Jiguang Liang, Xiaofei Zhou, Yue Hu, Li Guo, and Shuo Bai. Conr: A novel method for sentiment word identification. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1943–1946, 2014.

[26] Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. Automatic construction of a context-aware sentiment lexicon: An optimization approach. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 347–356, New York, NY, USA, 2011. ACM.

[27] Masoud Makrehchi and Mohamed S. Kamel. Extracting domain-specific stopwords for text classifiers. *Intelligent Data Analysis*, 21(1):39–62, 2017.

[28] Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of International Conference on Learning Representations*, 2013.

[29] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

[30] Arjun Mukherjee, Abhinav Kumar, Bing Liu, Junhui Wang, Meichun Hsu, Malu Castellanos, and Riddhiman Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640, 2013.

[31] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *CVPR*, pages 685–694, 2015.

[32] Myle Ott, Claire Cardie, and Jeffrey T Hancock. Negative deceptive opinion spam. In *Proceedings of NAACL-HLT*, pages 497–501, 2013.

[33] Myle Ott, Yejin Choi, Claire Cardie, and Jeffrey T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 309–319, 2011.

[34] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 115–124, 2005.

[35] Sungrae Park, Wonsung Lee, and Il-Chul Moon. Efficient extraction of domain specific sentiment lexicon with active learning. *Pattern Recogn. Lett.*, 56(C):38–44, 2015.

[36] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: global vectors for word representation. In *Empirical Methods in Natural Language Proceeding*, pages 1532–1543, 2014.

[37] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Expanding domain sentiment lexicon through double propagation. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence*, pages 1199–1204, 2009.

[38] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034 [cs]*, 2013.

[39] Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465, 2013.

[40] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, 2013.

[41] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1556–1566, 2015.

[42] Jian Tang, Zhaoshi Meng, Xuanlong Nguyen, Qiaozhu Mei, and Ming Zhang. Understanding the limiting factors of topic modeling via posterior contraction analysis. In *Proceedings of the 31st International Conference on Machine Learning*, pages 190–198, 2014.

[43] Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Trans. Inf. Syst.*, 21(4):315–346, 2003.

[44] Alper Kursat Uysal. An improved global feature selection scheme for text classification. *Expert Syst. Appl.*, 43(C):82–92, 2016.

[45] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M. Rush, Bart van Merrinboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv:1502.05698 [cs, stat]*, 2015.

[46] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 347–354, 2005.

[47] Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.

[48] Wenpeng Yin and Schtze Hinrich. Convolutional neural network for paraphrase identification. In *The 2015 Annual Conference of the North American Chapter of the ACL*, pages 901–911, 2015.

[49] Hongliang Yu, ZhiHong Deng, and Shiyingxue Li. Identifying sentiment words using an optimization-based model without seed words. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 855–859, 2013.

[50] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of 13th European Conference on Computer Vision*, pages 818–833, 2014.

[51] Matthew D. Zeiler, Graham W. Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *Proceedings of 2011 International Conference on Computer Vision*, 2011.

[52] Xiaolin Zheng, Zhen Lin, Xiaowei Wang, Kwei-Jay Lin, and Meina Song. Incorporating appraisal expression patterns into topic modeling for aspect and sentiment word identification. *Know.-Based Syst.*, 61(1):29–47, 2014.

[53] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929, 2016.

Table 1: Notation Table

Notation	Description
$X = [x_1, x_2, \ldots, x_n]$	a short text containing n words
$\mathbf{x}_i \in \mathbb{R}^d$	a d-dimensional word embedding of the i-th word
$\mathbf{W} \in \mathbb{R}^{h*d}$	a *filter* in a convolution operation applied to h continuous word embeddings
$\mathbf{s}_c$	a score vector that indicates the importance scores of words in a given text for class c

Table 2: Accuracy@k on sentiment word identification

Dataset	Method	Classification Accuracy	Top-1	Top-3	Top-5
MR	TF-IDF	N/A	22.78%	23.19%	22.87%
	TF-IDF-Softmax	73.92%	43.93%	34.10%	28.39%
	SalMap-MAX	N/A	29.55%	25.28%	21.38%
	SalMap-AVG	N/A	17.52%	12.71%	11.22%
	SV-MAX	**78.32%**	61.01%	40.04%	29.12%
	SV-AVG	76.87%	**66.80%**	**43.28%**	**30.43%**
SST	TF-IDF	N/A	25.14%	25.08%	23.55%
	TF-IDF-Softmax	75.04%	50.67%	37.51%	31.03%
	SalMap-MAX	N/A	35.67%	29.51%	23.83%
	SalMap-AVG	N/A	17.83%	14.81%	11.78%
	SV-MAX	**82.33%**	68.35%	42.90%	30.50%
	SV-AVG	81.23%	**71.24%**	**45.32%**	**31.25%**

Table 3: Precision, recall and F1 on sentiment word identification

Dataset	Method	Top-1			Top-3			Top-5		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
MR	TF-IDF	30.48%	5.12%	8.77%	29.85%	15.05%	20.01%	28.57%	24.00%	26.08%
	TF-IDF-Softmax	45.83%	23.93%	31.44%	34.72%	54.37%	42.38%	28.78%	75.10%	41.61%
	SalMap-MAX	16.75%	8.32%	11.11%	13.01%	19.40%	15.56%	11.32%	28.12%	16.13%
	SalMap-AVG	17.52%	8.66%	11.59%	12.71%	18.85%	15.18%	11.22%	27.72%	15.98%
	SV-MAX	59.65%	30.36%	40.24%	39.66%	60.54%	47.92%	28.69%	72.98%	41.19%
	SV-AVG	**66.93%**	**33.79%**	**44.91%**	**43.22%**	**65.44%**	**52.05%**	**30.32%**	**76.53%**	**43.43%**
SST	TF-IDF	34.22%	6.46%	10.87%	31.47%	17.82%	22.75%	28.44%	26.74%	27.62%
	TF-IDF-Softmax	51.41%	27.64%	35.94%	37.28%	60.13%	46.03%	30.81%	**82.79%**	44.90%
	SalMap-MAX	20.00%	9.16%	12.13%	13.79%	21.28%	16.85%	11.68%	29.66%	16.75%
	SalMap-AVG	17.83%	8.66%	11.66%	13.81%	20.59%	16.57%	11.78%	28.62%	16.69%
	SV-MAX	66.62%	34.53%	45.49%	42.56%	66.21%	51.82%	30.22%	78.33%	43.61%
	SV-AVG	**71.32%**	**36.71%**	**48.47%**	**45.32%**	**70.00%**	**55.02%**	**31.38%**	80.78%	**45.20**

Table 4: Top-10 sentiment words selected by our models and baselines

	Positive Words	Negative Words
TF-IDF	zone, good, ya, year, liked like, fun, slight, worth, fantastic	bad, movie, hate, time, year good, work, just, characters, films
TF-IDF-Softmax	film, best, love, performances, good, funny, fun, heart, work, performance	movie, bad, just, like, feels, plot, long, minutes, dull, thing
SalMap-MAX	not, best, heart, performances, funny, beautiful, beautifully, enjoyable, charming, beautiful	bad, too, no, not, seems, lack, worst, rather, instead, better, only
SV-MAX	good, funny, best, well, love, performances, fun, drama, great, family	too, not, no, bad, only, never, nothing, little, less, dull
SV-AVG	best, live, great, entertaining, good, fascinating, fun, beautifully, enjoyable, charming	bad, too, not, dull, no, boring nothing, mess, problem, lack, bland

Table 5: Precision, recall and F1 on Discrimination-related Words Identification

Method	Top-1			Top-3			Top-5		
	Precision	Recall	F1	Precision	Recall	F1	Precsion	Recall	F1
TF-IDF	11.33%	4.07%	5.99%	8.44%	9.10	8.76%	7.80%	14.01%	10.02%
TF-IDF-Softmax	82.00%	26.00%	39.61%	65.56%	62.63%	64.06%	49.87%	79.41%	61.26%
SalMap-MAX	50.84%	14.20%	22.21%	40.63%	34.05%	37.05%	32.12%	44.87%	37.44%
SV-MAX	98.99%	**33.98%**	50.60%	71.48%	73.62%	72.53%	**50.33%**	**86.41%**	**63.61%**
SV-AVG	**99.65%**	33.92%	**50.62%**	**72.80%**	**74.34%**	**73.56%**	50.14%	85.34%	63.17%

Table 6: Top-10 discrimination-related words selected by our models and baselines

	Sexism-related Words	Racism-related Words
TF-IDF	live, miss, sexy, kikme, cams, pornvideos, chat, pussy, freeporn, cybersex	described, opposes, pbuh, prophet, muhammad, christianity, selfie, mecca, racism, racist
TF-IDF-Softmax	sexism, women, men, feminism, horny, online, sexy, woman, pussy, kikme	racism, racist, ignorance, black, pure,forms, opposes, christianity, mecca, selfie
SalMap-MAX	sexism, women, join, live, online, cams, pussy, pornvideos, kikme, freeporn	racism, islam, racist, ignorance, prophet, pbuh, end, love, christianity, black
SV-MAX	sexism, women, men, woman, feminism,sexist, female, male, horny, sexy	racism, racist, black, white, religion, hate, education, america, isla, ignorance
SV-AVG	sexism, women, men, feminism, horny sexy, woman, pussy, chat, cybersex	racism, racist, islam, black, igonorance, opposes, religion, mecca, white, hate

Table 7: Discrimination-related words selected by SV-AVG.

Racism-related Words
• ***Idiots***^{5} like you can't even define ***#racism***^{4} or own your own but always calling ***Black***^{2} people ***racists***^{1} or ***hateful***^{3}.
• ***Hostility***^{1} between ***minority***^{2} ***races***^{4} always intrigue me. Who's the ***enemy***^{5} here? Is there even one? #369hong #nonoboy ***#racism***^{3}
• Imagine a ***#white***^{3} person say ***#Africa***^{1} is too full of ***brown***^{2} people or Arabia is too full of #muslims? ... ***#racism***^{5} ***#whitegenocide***^{4}
• Cards against humanity is definitely ***racist***^{1}, have 10 ***'white***^{3} cards. And then whoever is funniest gets a ***'black"***^{2} ***card***^{5} ***#racism***^{4}
• US cited for ***police***^{3} ***violence***^{1}, ***#racism***^{2} in scathing #UN human ***rights***^{5} review #BlackLivesMatter #policebrutality ***#torture***^{4}
Sexism-related Words
• The next time a ***stranger***^{2} ***man***^{4} asks me to smile, I'll punch him in the face! I ***swear***^{5} it ***#Sexism***^{1} ***#Rude***^{3} #StopSexualisingWomen
• ***Yeah***^{5} ***fuck***^{3} ***#sexism***^{2} Unexpected ***Job***^{4} Requirement Video Game Streamers Comfortable Wearing ***Bikini***^{1} Top
• 80% Telegraph readers: ok to call ***boys***^{5} "sissies" to "man up" or ***girls***^{4} studying ***"male"***^{3} subjects ***"lesbians"***^{1} ***#sexism***^{2}
• Shocking that many ***#women***^{4} are still experiencing ***gender***^{4} ***bias***^{3} at work and unacceptable levels of ***#sexism***^{2} in the ***#workplace***^{5} still exist
• @EverydaySexism as a ***female***^{1} Dr I'd ***obviously***^{5} ***wear***^{4} a mini ***skirt***^{3} and have to be a 'baby' Dr #juniordoctors ***#sexism***^{2}

Table 8: Discrimination-related phrases selected by SV-AVG with filter $h = 2$.

Racism-related Words
• Wetherspoons going back to the 50's with all white staff[3] and customers and black guy[1] taking abuse in toilets #nostalgia #racism[2]
• Imagine a #white person[3] say #Africa is too full of brown people or Arabia is too full of #muslims? ... #racism[2] #whitegenocide[1]
• People bashing #BoycottStarWarsVII would be the first ones screaming #racism of the cast[1] was all or even mostly white[2]. #irony[3] #Hypocrites
• #Elxn42 Funny how @USER knows what a real #Black[2] man[1] is, but not what real #Journalism is. #cdnpoli #tlot #tcot #humour #racism[3]
• Black Judge[2] Gives Armed Robbers Light Sentence Saying Three-Year-Old White Victim[3] is Racist[1]
Sexism-related Words
• Supporting #Women[2] in #Sales[3] means #NOT being aggressive on the phone or in person w/us Support #genderequity #NOT #bias #sexism[1]
• Shocking that many #women are still experiencing gender bias[1] at work and unacceptable levels of #sexism[2] in the #workplace[3] still exist
• 80% Telegraph readers: ok to call boys "sissies"[2] to "man up" or girls studying "male"[3] subjects "lesbians" #sexism[1]
• Yep, we really need Australian media outlets sharing whether a woman is wearing[2] a bra[3] while grocery shopping #sexism[1]
• You can make sexual[3] assault[2] prevention possible by speaking out against offensive jokes! #SAAM2016 #PreventionPossibleCC #SAAM #Rape #Sexism[1]

Table 9: Top-10 words from truthful and deceptive positive reviews selected by our model and baselines

	Deceptive Positive Words	Truthful Positive Words
TF-IDF	hotel, room, hilton, **great**, rock, really, east, family, hard, **loved**	hotel, great, really, large, service, suite, room, free, place, talbott
TF-IDF-Softmax	chicago, husband, visit, amazing, family, staying, looking, spa, vacation, **luxury**	floor, bathroom, location, large, small, rate, street, reviews, michigan, great
SalMap-MAX	experience, family, spa, husband, **luxury**, **luxurious**, vacation, wedding, wife, food	floor, large, small, reviews, rate, street, priceline, blocks, wife, upgraded
SV-MAX	hotel, room, **great**, **comfortable**, **like**, recommend, **amazing**, **luxury**, business, **clean**	floor, bathroom, large, small, location street, blocks, booked, river, parking

Table 10: Truthful and deceptive review phrases selected by SV-MAX with filter $h = 2$.

Truthful Positive Reviews	We travel to Chicago regularly and have always wanted to stay at the Hilton Chicago. We booked a PriceLine room[1] on the weekend of the half marathon. The place was busy[3], but the staff certainly found time for each individual guest. The room was nice and just what we expected. The pool area is also very nice for a downtown hotel. The location was great as we were taking in the Bears 49ers game on a Thursday night[2] and a Sox game on a Friiday night. We would certainly stay there again and would recomment it to others. Great quality even at the regular room price.
Deceptive Positive Reviews	I am often traveling on business and I always try to stay at the James. The modern design[3] of the lobby and restaurant with wood and glass creates a luxurious atmosphere[1] and the streamlined, comfortable room design give an upscale aesthetic[2] like that of a penthouse loft apartment. The availability of computers and coffee in the James' business centers means I do not have to lug my laptop around the hotel. The James provides the perfect setting for business travelers and is a cozy place to retire to in the evenings. The ease of breakfast service in the rooms means that I do not have to stress or rush, as my time is valuable. I love the James.

Figure 1. Given a text: *Ferrara's strongest and most touching movie of recent years*, the score vector $\mathbf{s}_c$ locates the task-specific words.

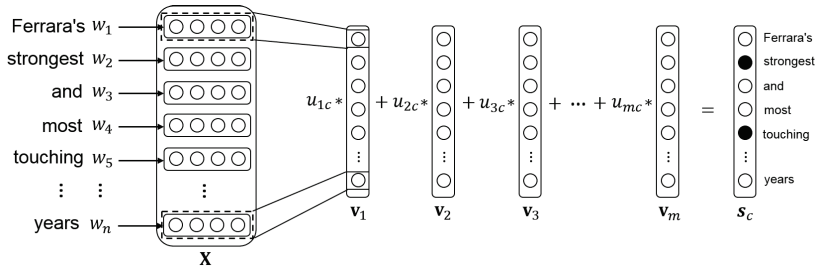

Figure 1: Given a text: *Ferrara's strongest and most touching movie of recent years*, the score vector $\mathbf{s}_c$ locates the task-specific words.