Social Network Analysis and Mining

Incorporating Pre-Training in Long Short-Term Memory Networks for Tweet Classification

--Manuscript Draft--

Manuscript Number:	SNAM-D-18-00032R2
Full Title:	Incorporating Pre-Training in Long Short-Term Memory Networks for Tweet Classification
Article Type:	Original Article
Corresponding Author:	Xintao Wu University of Arkansas Fayetteville UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Arkansas Fayetteville
Corresponding Author's Secondary Institution:	
First Author:	Shuhan Yuan
First Author Secondary Information:	
Order of Authors:	Shuhan Yuan
	Xintao Wu
	Yang Xiang
Order of Authors Secondary Information:	
Funding Information:	National Science Foundation (1646654) — Dr. Xintao Wu
	National Natural Science Foundation of China (71571136) — Dr. Yang Xiang
	Science and Technology Commission of Shanghai Municipality (CN) (14511108002,16JC1403000) — Dr. Yang Xiang
Abstract:	The paper presents deep learning models for tweet classification. Our approach is based on the Long Short-Term Memory (LSTM) recurrent neural network and hence expects to be able to capture long-term dependencies among words. We first focus on binary classification task. The basic model, called LSTM-TC, takes word embeddings as inputs, uses LSTM to derive the semantic tweet representation, and applies logistic regression to predict the tweet label. The basic LSTM-TC model, like other deep learning models, requires a large amount of well-labeled training data to achieve good performance. To address this challenge, we further develop an improved model, called LSTM-TC*, that incorporates a large amount of weakly-labeled data for classifying tweets. Finally, we extend the models, called LSTM-Multi-TC and LSTM-Multi-TC*, to multiclass classification task. We present two approaches of constructing the weakly-labeled data. One is based on hashtag information and the other is based on the prediction output of a traditional classifier that does not need a large amount of well-labeled training data. Our LSTM-TC* and LSTM-Multi-TC* models first learn tweet representation based on the weakly-labeled data, and then train the classifiers based on the small amount of well-labeled data. Experimental results show that: (1) the proposed methods can be successfully used for tweet classification and outperform existing state-of-the-art methods; (2) pre-training tweet representations, which utilizes weakly-labeled tweets, can significantly improve the accuracy of tweet classification.
Response to Reviewers:	See attachment.

Noname manuscript No.
(will be inserted by the editor)

Incorporating Pre-Training in Long Short-Term Memory Networks for Tweet Classification

Shuhan Yuan · Xintao Wu · Yang Xiang

Received: date / Accepted: date

Abstract The paper presents deep learning models for tweet classification. Our approach is based on the Long Short-Term Memory (LSTM) recurrent neural network and hence expects to be able to capture long-term dependencies among words. We first focus on binary classification task. The basic model, called LSTM-TC, takes word embeddings as inputs, uses LSTM to derive the semantic tweet representation, and applies logistic regression to predict the tweet label. The basic LSTM-TC model, like other deep learning models, requires a large amount of well-labeled training data to achieve good performance. To address this challenge, we further develop an improved model, called LSTM-TC*, that incorporates a large amount of weakly-labeled data for classifying tweets. Finally, we extend the models, called LSTM-Multi-TC and LSTM-Multi-TC*, to multiclass classification task. We present two approaches of constructing the weakly-labeled data. One is based on hashtag information and the other is based on the prediction output of a traditional classifier that does not need a large amount of well-labeled training data. Our LSTM-TC* and LSTM-Multi-TC* models first learn tweet representation based on the weakly-labeled data, and then train the classifiers based on the small amount of well-labeled data. Experimental results show that: (1) the proposed methods can be successfully used for tweet classification and outperform existing state-of-the-art methods; (2) pre-training tweet representations, which utilizes weakly-labeled tweets, can significantly improve the accuracy of tweet classification.

Keywords LSTM, tweets classification, pre-training, deep learning

Shuhan Yuan
University of Arkansas, Fayetteville, AR, USA
E-mail: sy005@uark.edu

Xintao Wu
University of Arkansas, Fayetteville, AR, USA
E-mail: xintaowu@uark.edu

Yang Xiang
Tongji University, Shanghai, China
E-mail: shxiangyang@tongji.edu.cn

1 Introduction

In recent years, online social network sites such as Twitter become important and influential as users actively share their stories and express their emotions or opinions there. The text classification in analyzing user-generated content (UGC) in social networks has become a hot research topic in the natural language processing area. Various types of classification tasks like sentiment analysis, sarcasm detection or political ideology detection have been investigated (Iyyer et al, 2014; Joshi et al, 2017; Tang et al, 2014). Recent work in natural language processing has shown that deep neural networks (Bengio et al, 2013; LeCun et al, 2015) can learn meaningful representations (or features) of text and use them to achieve high prediction accuracy in applications like sentiment analysis (Gambhir and Gupta, 2017; Irsoy and Cardie, 2014; Zhang et al, 2015). Deep learning models avoid hand-designed text features and learn text representations automatically by training on a large amount of well-labeled data. In reality, it is often difficult to obtain such a large amount of well-labeled data because manually labeling tweets is tedious for users. These deep learning approaches tend to face the problem of poor prediction accuracy when only an insufficient amount of well-labeled data is available. In this paper, we tackle this challenge by effectively incorporating weakly-labeled data in deep learning to improve the classification results.

Our basic tweet classification models, called LSTM-TC and LSTM-Multi-TC, are based on the Long Short-Term Memory (LSTM) recurrent neural network. LSTM has been proved to be able to capture long-term dependencies among words. Our LSTM-TC and LSTM-Multi-TC models sequentially take each word in a tweet as input, extract its salient information, and embed each word into a semantic vector. When the models reach the last word, the hidden layers of the LSTM networks provide semantic representations of the whole tweets. On top of the tweet representation, the LSTM-TC applies a logistic regression classifier to identify tweet class. Meanwhile, the LSTM-Multi-TC adopts a softmax classifier to predict the class of each tweet.

The basic LSTM-TC and LSTM-Multi-TC models, like other deep learning models, still require a large amount of training data to achieve good performance. We further develop improved models, called LSTM-TC* and LSTM-Multi-TC*, that incorporate a large amount of weakly-labeled tweets that are either available or can be easily constructed. Our LSTM-TC* and LSTM-Multi-TC* first learn the tweet representations based on a large amount of weakly-labeled data in the pre-training phase, and then train the classifier (logistic regression or softmax) based on the small amount of well-labeled data in the second phase. In general, given a weakly-labeled dataset with K classes, the tweets that are weakly-labeled as class k are more likely to be the truly class k than those weakly-labeled tweets belonging to other classes. For example, in the binary classification task, the tweets that are weakly-labeled as positive are more likely to be truly positive than those weakly-labeled negative tweets. Note that in binary classification, the weakly-labeled tweets are similar to one class of tweets and far from the other class of tweets. However, in multiclass classification, the weakly-labeled one class of tweets are far from the rest classes of tweets. Hence, we propose different strategies in LSTM-TC* and LSTM-Multi-TC* to evaluate the similarity between the weakly-labeled tweets and well-labeled tweets. The pre-training phase of LSTM-TC* or LSTM-Multi-TC* expects to learn better tweet representations by utilizing the similarity between the weakly-labeled tweets and well-labeled tweets, and hence LSTM-TC* and LSTM-Multi-TC* can achieve better performance.

We show two approaches to construct the weakly-labeled data for tweet classification. The first approach is based on hashtag information. In Twitter, users often add hashtags, which mark keywords or topics, in their tweets. We consider tweets containing specific hashtags which are related to the classification task are weakly-labeled positive tweets and tweets which do not contain specific hashtags are weakly-labeled negative tweets. For example, in research on sarcasm detection, the tweets containing hashtags *#sarcasm* or *#sarcastic* were marked as positive data (Bamman and Smith, 2015; Rajadesingan et al, 2015). However, those tweets are actually weakly-labeled positive because those hashtags may not exactly indicate the sarcasm of the underlying tweet. For example, the tweet "*#sarcasm is an important research in languages*" is not sarcastic. The second approach is based on the prediction results of a traditional classifier (e.g., Naive Bayes) that is first trained over the small amount of well-labeled data and then is used to predict a large amount of unlabeled data. Note that although the traditional classifier cannot achieve high prediction accuracy, it rarely requires a large amount of training data. There are some other approaches to construct the weakly-labeled data. We can adopt domain-specific lexicons to build weakly-labeled datasets. For example, in sentiment analysis area, the sentence with words in positive sentiment lexicon can be labeled as weakly-positive. We can also adopt the topic model to extract topic words from documents. The topic words can be used as a domain-specific lexicon. Meanwhile, the documents with the same topic distribution calculated by the topic model can be also considered in the same class.

To evaluate our proposed models, we focus on one case study — identifying discrimination-related tweets. Firstly, we label a small number of tweets as the golden dataset. Then, to compose the weakly-labeled dataset based on hashtags, we consider tweets containing hashtags like *#sexism* or *#racism* are weakly-labeled positive tweets and these tweets likely contain discrimination-related information. We consider tweets only containing hashtags like *#news* or *#breaking* as weakly-labeled negative tweets. Meanwhile, we also adopt the classification results of well-trained Naive Bayes on tweets as weakly-labeled data. We conduct a series of experiments over our crawled data and compare with traditional classifiers such as SVM, logistic regression, and Naive Bayes as well as the recurrent neural network. Experimental results show that: (1) the proposed methods can be successfully used for tweet classification and significantly outperform existing methods; (2) pre-training text representations, which utilizes weakly-labeled tweets, can improve classification accuracy; and (3) the quantity and quality of weakly-labeled tweets affect the prediction performance of LSTM-TC* and LSTM-Multi-TC*.

2 Tweet Classification

Using deep neural networks for text classification is usually formed by first training the latent representation

of text by non-linear transformations and then training the classifier based on text representations as inputs. In our experiments, we use LSTM to model the tweet representations and apply a classifier for prediction. We first revisit the LSTM in Section 2.1 and then present the basic LSTM-TC model for binary classification in Section 2.2. In Section 2.3, we then present our improved LSTM-TC* model that uses the weakly-labeled tweets to train LSTM for learning tweet representations and uses the well-labeled data to train the classifier and fine-tune the whole model. Finally, we extend our LSTM-TC and LSTM-TC* models to LSTM-Multi-TC and LSTM-Multi-TC* for multiclass classification in Section 2.4.

2.1 Long Shot-Term Memory Network Revisited

Recurrent Neural Networks (RNNs) are a class of deep neural networks and have been used extensively in time sequence modeling (Graves, 2013; Tomas et al, 2010). RNN, when used for sentence embedding, can find a dense and low dimensional semantic representation of a sentence by sequentially and recurrently processing each word and mapping it into a semantic vector. However, standard RNNs are difficult to train over long sequences of text because of gradient vanishing and exploding (Bengio et al, 1997). Long Shot-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) was proposed to model temporal sequences and capture their long-range dependencies more accurately than the standard RNNs.

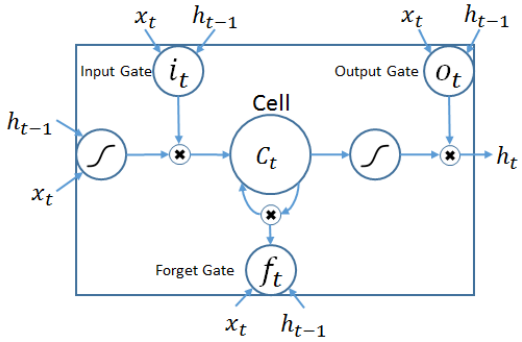

Fig. 1 Long short-term memory cell

LSTM contains special units called memory blocks in the recurrent hidden layer. Each memory block contains self-connected internal memory cells and special multiplicative units called gates to control the flow of information. Each memory block has an input gate that controls the flow of input activations into the memory cell, an output gate that controls the output of cell activations into the rest of the network, and a forget gate that allows the cells to forget or reset the cell's memory. These gates are used to control the flow of information through the internal states and allow the cell in LSTM memory block to store information over long time durations, thus avoiding the vanishing gradient problem. LSTM has various modifications. In our work, we adopt a widely used LSTM model (Gers et al, 1999). Figure 1 illustrates a single LSTM memory cell used in our work. The LSTM computes the hidden state $\mathbf{h_t} \in \mathbb{R}^{d_h}$ by Equation 1,

$$\begin{aligned}
\tilde{\mathbf{c}}_t &= \tanh(\mathbf{W_c}\mathbf{x}_t + \mathbf{U_c}\mathbf{h}_{(t-1)} + \mathbf{b_c}) \\
\mathbf{i}_t &= \sigma(\mathbf{W_i}\mathbf{x}_t + \mathbf{U_i}\mathbf{h}_{(t-1)} + \mathbf{b_i}) \\
\mathbf{f}_t &= \sigma(\mathbf{W_f}\mathbf{x}_t + \mathbf{U_f}\mathbf{h}_{(t-1)} + \mathbf{b_f}) \\
\mathbf{o}_t &= \sigma(\mathbf{W_o}\mathbf{x}_t + \mathbf{U_o}\mathbf{h}_{(t-1)} + \mathbf{b_o}) \\
\mathbf{c}_t &= \mathbf{i}_t \odot \tilde{\mathbf{c}}_t + \mathbf{f}_t \odot \mathbf{c}_{t-1} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned} \tag{1}$$

where $\mathbf{x}_t \in \mathbb{R}^{d_w}$ is the representation of the t-th word; σ is the sigmoid activation function; $\odot$ represents element-wise product; $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$, $\mathbf{c}_t$ indicate the input gate, forget gate, output gate, and cell activation vector; $\mathbf{h}$ is the hidden vector. We denote all parameters in LSTM as $\theta_1 = [\mathbf{W_i}, \mathbf{U_i}, \mathbf{b_i}, \mathbf{W_f}, \mathbf{U_f}, \mathbf{b_f}, \mathbf{W_o}, \mathbf{U_o}, \mathbf{b_o}, \mathbf{W_c}, \mathbf{U_c}, \mathbf{b_c}]$. Unlike the feed-forward multi-layer neural network, at time step t, the current hidden layer $\mathbf{h}_t$ gets feedback information from the previous hidden state $\mathbf{h}_{t-1}$ and new information from input $\mathbf{x}_t$ by using some nonlinear activation functions.

2.2 Basic LSTM-TC Model for Binary Classification

Tweet Representation. To use deep neural networks, we map each word w in a tweet to a d_w-dimensional real-valued semantic vector space $\mathbf{x} \in \mathbb{R}^{d_w}$. These word vectors are trained in an unsupervised way on a large text corpus and several approaches have been proposed to train the word embeddings (Bengio et al, 2003; Mikolov et al, 2013; Shen et al, 2015). Once the word vectors are well-trained, they can capture the hidden semantic and grammatical features of words. We model the tweet representation based on the semantic composition idea (Blacoe and Lapata, 2012; Mitchell and Lapata, 2010; Socher et al, 2012). The semantic composition aims to understand phrases and sentences by composing the meaning of each word through a composition function.

LSTM is able to model tweets with varied length by sequentially processing each word to a fixed length hidden state. For a tweet that contains n words, $(w_1, \cdots, w_n)$,

given the t-th word w_t of a tweet, the hidden state $\mathbf{h}_t \in \mathbb{R}^{d_h}$ is computed by Equation 1 using the current word representation $\mathbf{x}_t$ and the previous hidden state $\mathbf{h}_{t-1}$ as input. The LSTM computes a sequence of hidden vectors $\mathbf{h} = (\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n)$ by using the word vectors $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n)$ in the tweet as inputs one by one. We consider these hidden vectors capture the hidden semantic features of the tweet. At time step t, the hidden vector captures the semantic feature of the tweet until the t-th word. Finally, the model combines the hidden vector sequence by mean pooling operation to form the representation of the tweet $\mathbf{r} \in \mathbb{R}^{d_h}$:

$$\mathbf{r} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{h}_i. \tag{2}$$

Tweet Classification. We stack the logistic regression layer on top of the LSTM layer to classify tweets. The logistic regression layer takes the tweet representation of the i-th well-labeled tweet in the training dataset as input and predicts the label of tweet $\hat{y}_i$. The logistic regression function is:

$$P(\hat{y}_i|\mathbf{r}_i, \mathbf{u_l}, b_l) = \frac{1}{1 + e^{-(\mathbf{u}_l^T \mathbf{r}_i + b_l)}}, \tag{3}$$

where $\mathbf{r}_i$ is the representation of the i-th tweet, and $\mathbf{u}_l, b_l$ are the parameters of logistic regression. The architecture of the model is described in Figure 2. The model aims to minimize the cross entropy as loss function to optimize the model. The loss function is defined as:

$$\boldsymbol{L}_1(\hat{\mathbf{y}}; \theta_2) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(P(\hat{y}_i)) + (1 - y_i)\log(1 - P(\hat{y}_i)), \tag{4}$$

where N is the number of tweets in the training dataset, and $\theta_2 = [\mathbf{u_l}, b_l] \cup \theta_1$ is the parameters in LSTM and logistic regression.

Algorithm 1 shows the complete training method for tweet classification. We iterate our algorithm *Epoch* times. In each running, we first compute tweet representation r_i (Line 4), fit the logistic regression model (Lines 5-6), and update θ_2 (Line 7). The parameter θ_2 is optimized by Adadelta using back-propagation algorithm. Adadelta (Zeiler, 2012) is an advanced gradient-based approach, which provides a per-dimension learning rate. Adadelta can speed up the convergence rate compared to the traditional stochastic gradient descent (SGD) approach. We use dropout (Srivastava et al, 2014) as model regularizer. Dropout is an effective way to prevent neural networks from over-fitting by randomly dropping a portion of hidden units at the logistic regression layer during the training period. We also adopt mini-batch training (LeCun et al, 2012) to accelerate the training procedure.

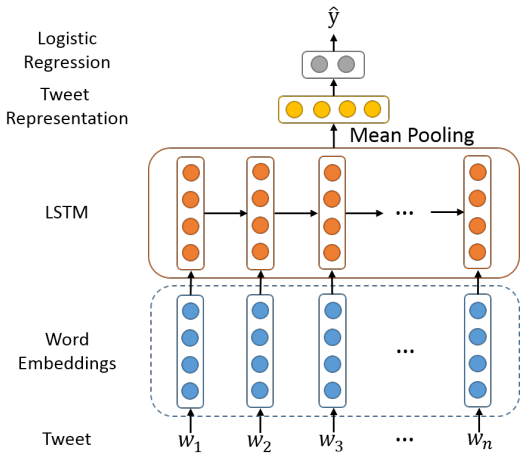

Fig. 2 LSTM-TC model for tweet binary classification

Algorithm 1: LSTM-TC: Basic Model for Tweet Classification

Inputs : Training dataset T
Maximum training epoch *Epoch*
Outputs: Trained Tweet Classification Model

```
1  j ← 0;
2  while j < Epoch do
3      for each tweet i in T do
4          compute r_i by Eq. 1,2 on T;
5          compute ŷ_i by Eq. 3 based on r_i;
6          compute L_1(ŷ_i; θ_2) by Eq. 4;
7          update θ_2 with Adadelta;
8      end
9      j ← j + 1;
10 end
```

2.3 Improved LSTM-TC* Model for Binary Classification

In general, training a deep neural network requires a large amount of training data. The basic LSTM-TC may not be applicable due to lacking a large amount of well-labeled training data in tweet classification tasks. Because the key to success of deep learning models is representation learning, we propose to train the tweet representation using the weakly-labeled data in our improved LSTM-TC* model. Algorithm 2 shows our LSTM-TC* model, which contains the pre-training phase (Lines 2-12) and the basic model training phase (Lines 13-22). In each phase, we iterate our algorithm with a fixed number of running. Similar to Algorithm 1, we optimize the parameters by Adadelta using back-propagation al-

gorithm and adopt dropout as model regularizer to prevent over-fitting and improve performance.

Formally, our training data has four parts: T^+ containing well-labeled positive tweets, T^- containing well-labeled negative tweets, $\tilde{T}^+$ containing weakly-labeled positive tweets, and $\tilde{T}^-$ weakly-labeled negative tweets.

Algorithm 2: LSTM-TC*: Improved Model for Binary Classification

Inputs : Well-labeled dataset T^+,T^-
Weakly-labeled dataset $\tilde{T}^+$,$\tilde{T}^-$
Maximum pre-training epoch *preEpoch*
Maximum training epoch *trainEpoch*
Outputs: Trained Classification Model

1 $j, m \leftarrow 0$;
2 compute $\mathbf{q}^+$, $\mathbf{q}^-$ by Eq. 1,2,5 on T^+,T^-;
3 **while** $j < preEpoch$ **do**
4 **for** *each tweet i in* $\tilde{T}^+$,$\tilde{T}^-$ **do**
5 compute $\tilde{\mathbf{r}}_i^+$,$\tilde{\mathbf{r}}_i^-$ by Eq. 1,2 on $\tilde{T}^+$, $\tilde{T}^-$;
6 compute $\boldsymbol{L}_2(\delta;\theta_1)$ by Eq. 9 on $\tilde{\mathbf{r}}_i^+$,$\mathbf{q}^+$,$\mathbf{q}^-$;
7 update θ_1 by Adadelta;
8 compute $\boldsymbol{L}_2(\delta;\theta_1)$ by Eq. 9 on $\tilde{\mathbf{r}}_i^-$,$\mathbf{q}^+$,$\mathbf{q}^-$;
9 update θ_1 by Adadelta;
10 **end**
11 $j \leftarrow j + 1$;
12 **end**
13 $T = shuffle(T^+, T^-)$;
14 **while** $m < trainEpoch$ **do**
15 **for** *each tweet i in* T **do**
16 compute $\mathbf{r}_i$ by Eq. 1,2 on T;
17 compute $\hat{y}_i$ by Eq. 3 based on $\mathbf{r}_i$;
18 compute $\boldsymbol{L}_1(\hat{y}_i;\theta_2)$ by Eq. 4;
19 update θ_2 by Adadelta;
20 **end**
21 $m \leftarrow m + 1$;
22 **end**

2.3.1 Pre-training Phase

In the pre-training phase, we train the LSTM to learn the semantic tweet representation based on the similarity between the weakly-labeled tweets and well-labeled tweets. As discussed in the introduction, the weakly-labeled positive tweets are likely to be truly positive than those weakly-labeled negative tweets. Therefore, the representation of a weakly-labeled positive tweet is similar to the *representation of positive class* and far to the *representation of negative class*.

We first use the LSTM to build $\mathbf{q}^+$ the representation of positive class and $\mathbf{q}^-$ the representation of negative class by using T^+ and T^- as inputs, respectively.

For the i-th tweet in T^+, we first compute the tweet representation $\mathbf{r}_i^+$ using Equations 1 and 2 and then compute the representation of positive class:

$$\mathbf{q}^+ = \frac{1}{|T^+|}\sum_{i=1}^{|T^+|}\mathbf{r}_i^+, \tag{5}$$

where $|T^+|$ is the number of tweets in T^+. Similarly, we compute the representation of negative class $\mathbf{q}^-$ on T^-. $\mathbf{q}^+$ and $\mathbf{q}^-$ are centroids of representations of tweets in T^+ and T^- respectively.

We use cosine function to measure the similarity between the weakly-labeled positive tweet representation $\tilde{\mathbf{r}}_i^+$ and positive representation $\mathbf{q}^+$.

$$sim(\tilde{\mathbf{r}}_i^+, \mathbf{q}^+) = \frac{\tilde{\mathbf{r}}_i^+ \cdot \mathbf{q}^+}{\|\tilde{\mathbf{r}}_i^+\|\|\mathbf{q}^+\|}, \tag{6}$$

where $\|\cdot\|$ denotes the L2 norm. The similarity score between the weakly-labeled positive tweet representation $\tilde{\mathbf{r}}_i^+$ and negative representation $\mathbf{q}^-$ is denoted as $sim(\tilde{\mathbf{r}}_i^+, \mathbf{q}^-)$. Because the model computes the similarity pairwisely, we adopt the pairwise learning loss function proposed by (Yih et al, 2011) to train the model.

For the weakly-labeled positive tweet representation $\tilde{\mathbf{r}}_\mathbf{i}^+$, we define

$$\delta = sim(\tilde{\mathbf{r}}_i^+, \mathbf{q}^+) - sim(\tilde{\mathbf{r}}_i^+, \mathbf{q}^-). \tag{7}$$

In contrast, for the weakly-labeled negative tweet representation $\tilde{\mathbf{r}}_\mathbf{i}^-$, we define

$$\delta = sim(\tilde{\mathbf{r}}_i^-, \mathbf{q}^-) - sim(\tilde{\mathbf{r}}_i^-, \mathbf{q}^+). \tag{8}$$

Then, we use logistic loss function over δ to train the parameters:

$$\boldsymbol{L}_2(\delta;\theta_1) = \log(1 + \exp(-\gamma\delta)) \tag{9}$$

where θ_1 is the parameters of LSTM. By updating θ_1 based on Equations 7 and 8 for $\tilde{\mathbf{r}}_i^+$ and $\tilde{\mathbf{r}}_i^-$ respectively, we let the weakly-labeled positive tweet representation $\tilde{\mathbf{r}}_i^+$ close to the $\mathbf{q}^+$ and far away from the $\mathbf{q}^-$ and the weakly-labeled negative tweet representation $\tilde{\mathbf{r}}_i^-$ close to the $\mathbf{q}^-$ and far away from the $\mathbf{q}^+$.

Because the similarity function is the cosine function, $\delta \in [-2, 2]$. In order to have a larger range and penalize more on Equation 9, we use γ as a scaling function. Once the LSTM is pre-trained by weakly-labeled data (Lines 2-12), we can get semantic representations of tweets. We can further use the basic model to detect the labels of tweets (Lines 13-22). At this time, the small set of well-labeled data is used for training the classifier and fine-tuning the whole model.

2.4 LSTM-Multi-TC and LSTM-Multi-TC* Models for Multiclass Classification

In Section 2.2 and 2.3, we focus on the task of tweets binary classification based on LSTM with well-labeled and weakly-labeled data. In reality, multiclass classification task is more common than the binary classification task. In this section, we extend our existing models to tweet multiclass classification models (LSTM-Multi-TC and LSTM-Multi-TC*). The task is to predict the class label of a tweet given a well-labeled training dataset with K classes $T = \{T_1, \ldots, T_K\}$ and the corresponding weakly-labeled dataset $\tilde{T} = \{\tilde{T}_1, \ldots, \tilde{T}_K\}$.

LSTM-Multi-TC Model. We first extend the LSTM-TC model to LSTM-Multi-TC model. We build the tweet representation based on Equation 1 and 2. Then, we replace the logistic function with softmax function:

$$P(\hat{y}_i|\mathbf{r}_i, \mathbf{W}, \mathbf{b}) = \frac{\exp(\mathbf{w}_k^T \mathbf{r}_i + b_k)}{\sum_{k'=1}^{K} \exp(\mathbf{w}_{k'}^T \mathbf{r}_i + b_{k'})}, \tag{10}$$

where $\mathbf{r}_i$ is the representation of the i-th tweet, and $\mathbf{W}$, $\mathbf{b}$ are the parameters of softmax. We adopt the cross entropy as the loss function to train LSTM-Multi-TC:

$$\boldsymbol{L}_3(\hat{\mathbf{y}}; \theta_3) = -\sum_{i=1}^{N}\sum_{k=1}^{K} 1\{y_i = k\} \log P(\hat{y}_i|\mathbf{r}_i, \mathbf{W}, \mathbf{b}), \tag{11}$$

where N is the number of tweets, K is the number of classes, and $\theta_3 = [\mathbf{W}, \mathbf{b}] \cup \theta_1$ is parameters in LSTM and softmax.

LSTM-Multi-TC* Model. We further extend the LSTM-TC* model to LSTM-Multi-TC* model, which adopts the weakly-labeled data to pre-train the LSTM model.

In the pre-training phase, we train LSTM based on the similarity between weakly-labeled tweets and well-labeled tweets. Given a dataset with K classes, the representation of a weakly-labeled tweet in class k is similar to the representation of tweets in T_k and far away from the representation of tweets in T_{-k} in the well-labeled dataset. Thus, similar to LSTM-TC*, we define the representation of positive class $\mathbf{q}_k^+$ as the representation of class T_k and the representation of negative class $\mathbf{q}_{-k}^-$ as the representation of classes T_{-k}.

Unlike the binary classification, the representation of negative class in multiclass classification contains $K-1$ classes. We cannot just use mean operation in Equation 5 to combine all the tweet representations in different classes because not all the tweets contribute equally to the representation of the negative class. Therefore, we adopt the neural attention model (Bahdanau et al, 2015; Rush et al, 2015; Yang et al, 2016) to evaluate the importance of each tweet representation to the representation of positive (negative) class. The basic idea of attention model is to assign weight to the lower level representation in each position when computing the upper level representation. In this work, we aim to learn the representation of positive (negative) class from the representation of each tweet in that positive (negative) class. We let the model learn which tweet play a more important role to the meaning of positive (negative) class with the neural attention model, not just assigning equal weights to all tweets.

Given a weakly-labeled tweet in class $\tilde{T}_k$, the corresponding representation of negative class is computed based on the neural attention model:

$$\boldsymbol{s}_i = \tanh(\mathbf{W}_a \boldsymbol{r}_i + \boldsymbol{b}_a), \tag{12}$$

$$\alpha_i = \frac{\exp(\boldsymbol{u}_a^T \boldsymbol{s}_i)}{\sum_{i=1}^{|T_{-k}|} \exp(\boldsymbol{u}_a^T \boldsymbol{s}_i)}, \tag{13}$$

$$\boldsymbol{q}_{-k}^- = \sum_{i=1}^{|T_{-k}|} \alpha_i \boldsymbol{s}_i, \tag{14}$$

where $\mathbf{W}_a$, $\boldsymbol{b}_a$, $\boldsymbol{u}_a$ are the parameters of the neural attention model and $|T_{-k}|$ is the total number of tweets in classes T_{-k}.

In the attention model, we first feed each tweet representation $\boldsymbol{r}_i$ in T_{-k} to a one-layer neural network to compute the $\boldsymbol{s}_i$ as the hidden representation of $\boldsymbol{r}_i$. After obtaining all the hidden representation $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{|T_{-k}|}\}$, the weight of the i-th tweet to the representation of negative class is calculated through a softmax function. Then, we obtain the weights of tweets in the negative class $\{\alpha_1, \ldots, \alpha_{|T_{-k}|}\}$. Finally, the representation of negative class $\boldsymbol{q}_{-k}^-$ is computed as a weighted sum of each tweet representation and the corresponding weight. Figure 3 shows how to build the representation of negative class based on the neural attention model. We also follow the same procedure to generate the representation of positive class $\boldsymbol{q}_k^+$.

The advantage of the neural attention model is that it can dynamically assign a weight to each tweet representation according to its relatedness with the corresponding class. Especially, in our work, each tweet can be either in positive class or negative class and the weight of each tweet may be different in each class. Thus, using the neural attention model, we can get a more reasonable representation of the positive or negative class. Furthermore, because the objective of pre-training phase is based on the similarity between weakly-labeled tweets and well-labeled tweets, we expect the LSTM model can be pre-trained well by providing a good representation of the positive or negative class.

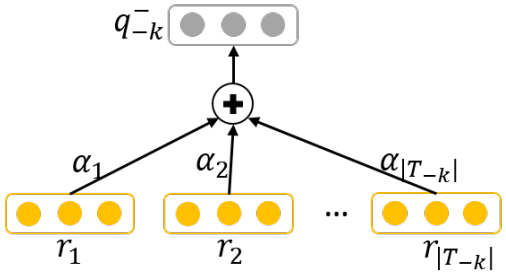

Fig. 3 Using the neural attention model to compose the representation of negative tweet class

The similarity scores between the representation of weakly-labeled tweet $\tilde{\mathbf{r}}_i$ in $\tilde{T}_k$ and the representation of positive and negative classes calculated by Equation 6 are denoted as $sim(\tilde{\mathbf{r}}_i, \boldsymbol{q}^+_k)$ and $sim(\tilde{\mathbf{r}}_i, \boldsymbol{q}^-_{-k})$. Then, we use logistic loss function to train the parameters:

$$\delta = sim(\tilde{\mathbf{r}}_i, \boldsymbol{q}^+_k) - sim(\tilde{\mathbf{r}}_i, \boldsymbol{q}^-_{-k}), \quad (15)$$

$$\boldsymbol{L}_4(\delta;\theta_4) = \log(1+\exp(-\gamma\delta)), \quad (16)$$

where $\theta_4 = [\mathbf{W}_a, \boldsymbol{b}_a, \boldsymbol{u}_a] \cup \theta_1$ is the parameters of LSTM and the neural attention model. Algorithm 3 shows the LSTM-Multi-TC* model. The LSTM and the neural attention model are trained together by iterating through all the classes in the weakly-labeled data (Lines 2-12). After that, we use the basic LSTM-Multi-TC model to predict the classes of tweets (Lines 13-22). Similar to Algorithm 1 and 2, we optimize the parameters by Adadelta and adopt dropout as model regularizer to prevent over-fitting and improve performance.

Algorithm 3: LSTM-Multi-TC*: improved model for tweet multiclass classification

Inputs : Well-labeled dataset $T = \{T_1,\ldots,T_K\}$
Weakly-labeled dataset $\tilde{T} = \{\tilde{T}_1,\ldots,\tilde{T}_K\}$
Maximum pre-training epoch *preEpoch*
Maximum training epoch *trainEpoch*
Outputs: Trained Multiclass Classification Model

1 $j, m \leftarrow 0$;
2 **while** $j < preEpoch$ **do**
3 **for** *each class* $\tilde{T}_k$ *in* $\tilde{T}$ **do**
4 **for** *each tweet* i *in* $\tilde{T}_k$ **do**
5 compute $\tilde{\mathbf{r}}_i$ by Eq. 1,2 on $\tilde{T}_k$;
6 compute $\boldsymbol{q}^+_k$, $\boldsymbol{q}^-_{-k}$ by Eq. 12,13,14 on T_k and T_{-k} ;
7 compute $\boldsymbol{L}_4(\delta;\theta_4)$ by Eq. 16 on $\tilde{\mathbf{r}}_i$, $\boldsymbol{q}^+_k$, $\boldsymbol{q}^-_{-k}$;
8 update θ_4 by Adadelta;
9 **end**
10 **end**
11 $j \leftarrow j+1$;
12 **end**
13 $T = shuffle(T_1,\ldots,T_K)$;
14 **while** $m < trainEpoch$ **do**
15 **for** *each tweet* i *in* T **do**
16 compute $\mathbf{r}_i$ by Eq. 1,2 on T;
17 compute $\hat{y}_i$ by Eq. 10 based on $\mathbf{r}_i$;
18 compute $\boldsymbol{L}_3(\hat{y}_i;\theta_3)$ by Eq. 11;
19 update θ_3 by Adadelta;
20 **end**
21 $m \leftarrow m+1$;
22 **end**

3 Experiments

In this section, we evaluate our methods for tweet classification and sentiment analysis. In particular, for tweet classification, we focus on identifying discrimination-related tweets.

3.1 Setup

Word Embeddings and Hyperparameters. Word embeddings are required as input in our framework. We use the off-the-shelf pre-trained word embeddings [1] provided by Mikolov (Mikolov et al, 2013). These word embeddings are widely used and have been shown to achieve good performance on many NLP tasks. We further randomly initialize the words which don't have pre-trained word embeddings but appear at least 5 times. The dimension of each word embedding is 300. The dimension of the hidden layer in LSTM is equal to the dimension of word embeddings. The pre-training epoch is 30 with early stopping. The training epoch is 100 with early stopping. The batch size is 30. The dropout rate is set to 0.5. γ in Equation 9 is set to 10.

Baselines. We compare our methods with the following baselines, logistic regression **(LR)**, support vector machine (**SVM**), Naive Bayes classifier (**NB**), eXtreme Gradient Boosting (**XGBoost**) (Chen and Guestrin, 2016), and convolutional neural network (**CNN**). We follow the approach proposed in (Kim, 2014) to build the convolutional neural network. The first four classifiers are trained on traditional N-gram features, in particular, unigram and bigram features. CNN takes word embeddings as inputs. We implement all deep neural networks using the Theano package (Bastien et al, 2012) and train the SVM, Naive Bayes, and logistic regression models using the Scikit-learn package [2]. We adopt the XGBoost package [3] to evaluate its performance.

Code. The complete source code and crawled data are available at `https://bitbucket.org/bookcold/pretraining_lstm`.

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

[2] `http://scikit-learn.org/stable/`

[3] `http://xgboost.readthedocs.io/en/latest/`

3.2 Tweet Classification

3.2.1 Tweet Binary Classification

We first evaluate our LSTM-TC and LSTM-TC* models to identify discrimination related tweets and non-discrimination related tweets.

Dataset. Because there is no benchmark dataset used for discrimination discovery, we built our dataset by crawling all tweets containing hashtags such as *#everydaysexism*, *#blackoncampus*, *#sexism*, *#racism*, *#racist*, and *#news* during the period from Nov 12 to Dec 31, 2015. We also crawled two specific web sites [4] which provide instances of discrimination stories. To build the training dataset T, we manually labeled and included those tweets truly describing the discriminatory phenomenon or sharing discriminatory stories in the positive golden dataset T^+ from tweets containing hashtags *#blackoncampus* or *#everydaysexism*. The negative golden dataset T^- contains (1) tweets that are not discrimination-related (although containing hashtags like *#blackoncampus*), and (2) tweets containing hashtag *#news*. The golden training dataset T contains 600 well-labeled tweets, $|T^+| = 300$ and $|T^-| = 300$.

To evaluate the performance of LSTM-TC* model, we build two datasets, $\tilde{T}_1$ and $\tilde{T}_2$, as weakly-labeled training data. The dataset $\tilde{T}_1$ is based on the hashtags of tweets. We consider tweets containing hashtags like *#sexism* or *#racism* are weakly-labeled discrimination tweets and these tweets likely contain discrimination-related information. One example is "*Why are female cabinet members suspect but male ones are not? #bias #sexism*". However, we would emphasize again that not all tweets containing such hashtags can be considered as discrimination-related. For instance, the tweet "*I study #sexism in my behavior research project*" is not discrimination-related. We select top 5000 with highest favorite counts as $\tilde{T}_1^+$. A tweet with a high favorite count number usually indicates high quality. Similarly, we collect tweets having *#news* hashtag and select top 5000 with the highest favorite counts as $\tilde{T}_1^-$. The dataset $\tilde{T}_2$ is based on the classification results of the Naive Bayes classifier. We first use T to train the Naive Bayes classifier and then apply it to predict each tweet in $\tilde{T}_1$ as either positive or negative. The positive (negative) tweets predicted by the Naive Bayes classifier are then treated as weakly-labeled discrimination tweets $\tilde{T}_2^+$ ($\tilde{T}_2^-$).

In our dataset, we only keep the tweets which contain at least five words. We tokenize each tweet by TweetNLP [5] and remove all the tokens beginning with @ symbol and urls. We also remove special hashtags such as *#everydaysexism* and *#blackoncampus* from each tweet as those hashtags may provide hints about the tweet's class.

Results. All the baseline classifiers and our LSTM-TC require only the well-labeled data T as input. In addition to T, our LSTM-TC* also has a weakly-labeled dataset $\tilde{T}$ in its input. In particular, LSTM-TC* (hashtags) uses the weakly-labeled data $\tilde{T}_1$ based on hashtag information and LSTM-TC* (NB) uses the weakly-labeled data $\tilde{T}_2$ based on the output of Naive Bayes. To evaluate the performance of different sizes of training data, we split T into training part and test part with different ratios. In particular, we set the size of training part as 120, 240, 360, and 480, and accordingly the size of test part as 480, 360, 240, and 120. We use 5-fold cross-validation to evaluate the classification performance and adopt accuracy as evaluation metrics in all our experiments.

Table 1 shows experimental comparisons of our proposed methods and baselines. We have the following observations. First, deep learning models shown in the last four rows significantly outperform those traditional classifiers that use hand-design features (N-grams). This improvement could be due to the use of word embeddings and the power of deep learning models. Second, two LSTM-TC* models are better than the basic model LSTM-TC, which demonstrates the incorporation of weakly-labeled data in the pre-training phase of LSTM does improve the prediction accuracy. Third, the LSTM-TC* (NB) achieves noticeable better performance than the LSTM-TC* (hashtags), which shows the quality of weakly-labeled data makes difference.

Table 1 Experimental results on discrimination related tweets detection as a binary classification task

Method	Number of well-labeled data			
	120	240	360	480
LR (unigrams)	0.762	0.780	0.791	0.823
LR (bigrams)	0.722	0.774	0.794	0.840
SVM (unigrams)	0.520	0.521	0.725	0.713
SVM (bigrams)	0.659	0.736	0.756	0.765
NB (unigrams)	0.764	0.827	0.839	0.860
NB (bigrams)	0.770	0.852	0.875	0.870
XGBoost (unigrams)	0.843	0.841	0.825	0.833
XGBoost (bigrams)	0.850	0.827	0.817	0.817
CNN	0.862	0.871	0.884	0.895
LSTM-TC	0.860	0.873	0.878	0.900
LSTM-TC* (hashtags)	0.864	0.889	0.892	0.902
LSTM-TC* (NB)	**0.906**	**0.918**	**0.928**	**0.933**

Effect of the size of the weakly-labeled dataset. We use different sizes of the weakly-labeled dataset to

[4] `http://everydaysexism.com/`, `http://stemfeminist.com/`

[5] `http://www.cs.cmu.edu/~ark/TweetNLP/`

analyze their effect on the accuracy of discrimination prediction. We reduce the number of weakly-labeled tweets to 8000, 6000, 4000 and 2000. In each dataset, the number of the weakly-labeled discrimination-related tweets equals to the weakly-labeled non-discrimination-related tweets. The size of the well-labeled training data is 240 and that of the well-labeled test data is 360. Figure 4 shows how prediction accuracy changes with the size of the weakly-labeled data for both LSTM-TC* (hashtags) and LSTM-TC* (NB). One observation is that the decrease of the weakly-labeled data reduces the prediction accuracy of both methods, which is expected. One surprising result is that the LSTM-TC* (NB) even with the size as 2000 achieves better prediction accuracy than the LSTM-TC* (hashtags) with the size as 8000. It indicates the importance of the quality of the weakly-labeled data. In our future work, we will study how to construct the high-quality weakly-labeled data by combining multiple classifiers or better using hashtag information. We will also study whether the truly large number of weakly-labeled tweets can increase the prediction accuracy when we crawl more tweets for our experiments.

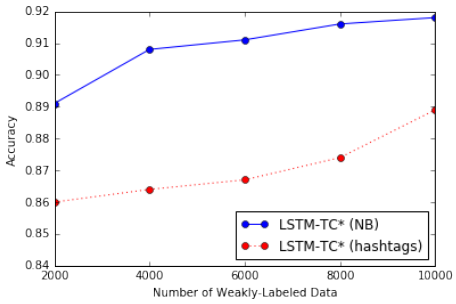

Fig. 4 Effect of the size of the weakly-labeled dataset

Model Analysis. We investigate how the pre-training on the weakly-labeled dataset affects the final prediction results. We compare our pre-trained LSTM with the random initialization LSTM for modeling the representations of tweets in the golden dataset. Figures 5 and 6 show the visualization of tweets representations in the training dataset computed by random initialization LSTM and our pre-trained LSTM in LSTM-TC* (NB) with 10000 weakly-labeled data. The vertical axis is the index number of each tweet in the golden training dataset (T^+ in left figures and T^- in right figures) and the horizontal axis shows each dimension of tweet representation. Each row then corresponds to a 300-dimension vector value of one particular tweet's representation. We use color to describe the value of tweet representation. We can see that the colors shown in Figure 5 (a) and Figure 5 (b) are very close. It means that the representations of discrimination-related and non-discrimination related tweets computed by the random initialized LSTM are similar. In contrast, the colors shown in Figure 6 (a) and Figure 6 (b) are significantly different. It demonstrates that our LSTM pre-training can map different classes of tweets to different regions in the projected multi-dimensional feature space.

3.2.2 Tweet Multiclass Classification

We evaluate our LSTM-Multi-TC and LSTM-Multi-TC* models for tweet multiclass classification. We train the models to detect the sexism-related, racism-related and non-discrimination tweets.

Dataset. We adopt the similar strategy to build the dataset for multiclass classification. The golden training dataset T contains sexism-related tweets $|T_s| = 300$, racism-related tweets $|T_r| = 300$, and non-discrimination related tweets $|T_n| = 300$. We only build one weakly-labeled dataset $\tilde{T}$ based on the classification results of the Naive Bayes classifier on the tweets containing hashtags *#sexism*, *#racism*, and *#news*. This is because the LSTM-TC* (NB) achieves much better results than the LSTM-TC* (hashtags) for tweets binary classification. For each class in the weakly-labeled dataset, the number of tweets is 4000.

Result. We evaluate the performance of different models on different sizes of training data. We adopt the same splitting ratio as the previous experiment to split the golden dataset T into training part and test part and use 5-fold cross-validation to evaluate the classification performance.

Table 2 shows experimental results of our proposed models and baselines. The overall results are similar to the results of binary tweet classification. LSTM-Multi-TC* outperform the baselines. Meanwhile, LSTM-Multi-TC* (NB) achieves noticeable better performance than the basic LSTM-Multi-TC, which shows the incorporation of weakly-labeled data in the pre-training phase of LSTM with the neural attention model for multiclass classification does improve the prediction accuracy.

3.3 Sentiment Analysis

3.3.1 Binary Sentiment Analysis

Dataset. We adopt the Stanford Sentiment Treebank (SST) dataset with binary labels (positive and negative). SST is a movie review dataset and provides train/dev/test splits. The average length of sentences in the dataset is 19. In experiments, we adopt a subset of training data as golden training dataset T and use the rest of training data to build the weakly-labeled dataset

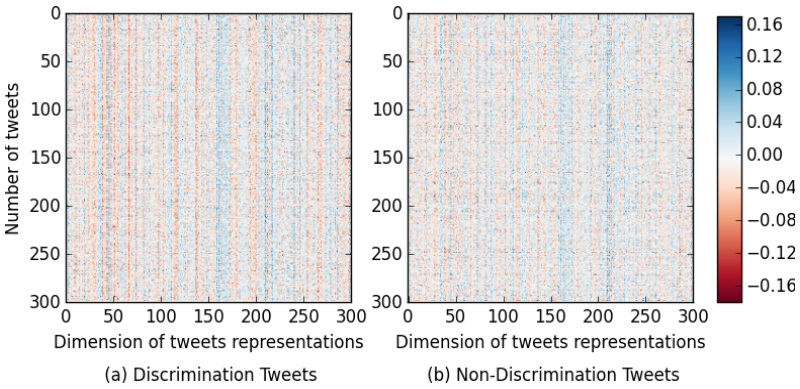

Fig. 5 Visualization of the tweet representation computed by randomly initialized LSTM

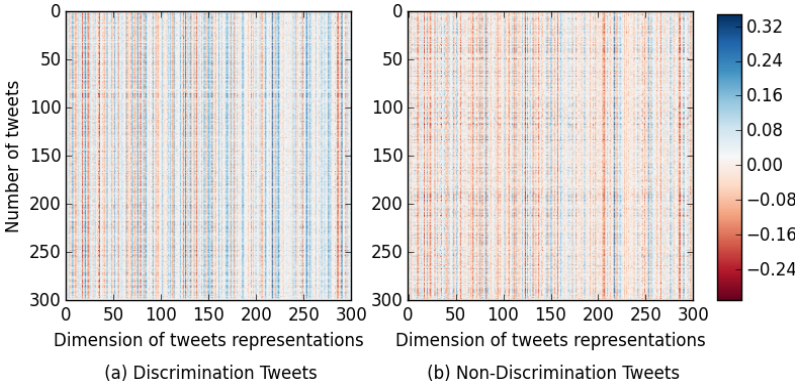

Fig. 6 Visualization of the tweet representation computed by our pre-trained LSTM

Table 2 Experimental results on discrimination related tweets detection as a multiclass classification task

Method	Number of well-labeled data			
	180	360	540	720
LR (unigrams)	0.7617	0.8070	0.8250	0.8400
LR (bigrams)	0.7425	0.8078	0.8250	0.8170
SVM (unigrams)	0.7589	0.8067	0.8266	0.8355
SVM (bigrams)	0.7400	0.8089	0.8283	0.8170
NB (unigrams)	0.7503	0.8092	0.8405	0.8400
NB (bigrams)	0.7483	0.8081	0.8289	0.8420
XGBoost (unigrams)	0.8000	0.8259	0.8511	0.8722
XGBoost (bigrams)	0.8000	0.8222	0.8538	0.8879
CNN	0.8069	0.8296	0.8111	0.8511
LSTM-Multi-TC	0.8042	0.8226	0.8495	0.8500
LTSM-Multi-TC* (NB)	**0.8508**	**0.8652**	**0.8800**	**0.8956**

$\tilde{T}$. In particular, the weakly-labeled positive (negative) dataset contains 80% of positive (negative) data and 20% of negative (positive) data. We report the average accuracy and F1 score after evaluating on the test dataset with 5 runs. For each deep learning model, we report the mean and standard deviation of accuracy and F1 score after 5 runs. Since the standard deviations of traditional classifiers are extremely small, we omit the results for clarity.

Results. We first evaluate the performance of LSTM-TC* on different sizes of golden training data. Table 3 shows the accuracies and F1 scores of LSTM-TC* and baselines for binary sentiment analysis. We can observe that the deep learning based approaches (CNN, LSTM-TC and LSTM-TC*) achieve much better performance than the traditional classifiers like SVM, Naive Bayes, logistic regression, and XGBoost. Meanwhile, both LSTM-TC and LSTM-TC* achieve higher accuracies and F1 scores than those of CNN. Furthermore, LSTM-TC* achieves higher accuracies and F1 scores than those of LSTM-TC, which shows the advantage of the incorporation of weakly-labeled data in the pre-training phase of LSTM.

We then evaluate the effect of the size of the weakly-labeled dataset. We use different sizes of weakly-labeled datasets to analyze the effects on binary sentiment analysis. From Figure 7, we can observe that using weakly-labeled dataset can improve the performance of binary classification when the well-labeled data is limited. Meanwhile, increasing the size of the weakly-labeled dataset can improve the classification performance. In particular, when the number of weakly-labeled data increases from 200 to 800, both accuracies and F1 scores have significantly increased.

Traditional classifiers with word embeddings as inputs. The deep neural networks take the word embeddings as inputs and compose the text representa-

Table 3 Experimental results on binary sentiment analysis with various sizes of golden training datasets

Method	Accuracy				F1			
	Number of well-labeled data				Number of well-labeled data			
	200	500	800	1000	200	500	800	1000
LR (unigrams)	0.5002	0.5024	0.5041	0.5058	0.6657	0.6647	0.6657	0.6669
LR (bigrams)	0.4997	0.4981	0.4986	0.4986	0.6661	0.6649	0.6649	0.6652
SVM (unigrams)	0.4997	0.5019	0.5024	0.5047	0.6663	0.6646	0.6649	0.6661
SVM (bigrams)	0.4997	0.4975	0.4991	0.4997	0.6666	0.6644	0.6649	0.6656
NB (unigrams)	0.4986	0.5030	0.5118	0.5173	0.6654	0.6667	0.6689	0.6716
NB (bigrams)	0.5002	0.5123	0.5277	0.5354	0.6659	0.6681	0.6735	0.6756
XGBoost (unigrams)	0.5326	0.5733	0.5738	0.5727	0.5999	0.4324	0.6832	0.6602
XGBoost (bigrams)	0.5310	0.5755	0.5277	0.5804	0.5890	0.4850	0.6449	0.6327
CNN	0.5853 ± 0.0155	0.6484 ± 0.0129	0.6846 ± 0.0014	0.6863 ± 0.0078	0.5933 ± 0.0364	0.6488 ± 0.0305	0.6943 ± 0.03150	0.6938 ± 0.0251
LSTM-TC	0.7092 ± 0.0054	0.7594 ± 0.0067	0.7692 ± 0.0074	0.7754 ± 0.0038	0.7049 ± 0.0178	**0.7636± 0.0097**	0.7716 ± 0.0076	0.7759 ± 0.0082
LSTM-TC*	**0.7542±0.0174**	**0.7661±0.0196**	**0.7771±0.0035**	**0.7808±0.0049**	**0.7549±0.0168**	0.7604±0.0125	**0.7730±0.0186**	**0.7789±0.0071**

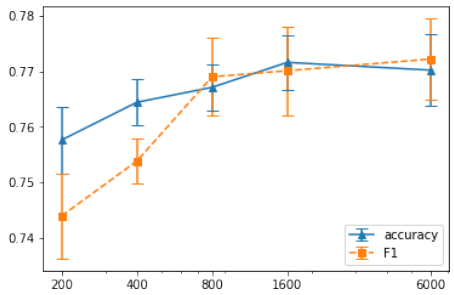

Fig. 7 Effect of the size of the weakly-labeled dataset

Table 4 Experimental results of traditional classifiers with word embeddings as inputs

	Accuracy	F1
LR	0.5013	0.6664
SVM	0.492	0.6659
XBGoost	0.4997	0.6656
LSTM-TC	0.7754	0.7759

tions. The output layers of deep neural networks adopt text representations for classification. Since the traditional classifiers cannot compose the text representations, we simply average the word embeddings over the text as the text representations. Table 4 shows the classification results of traditional classifiers and LSTM-TC. All the models are trained on 1000 well-labeled data. We can observe that LSTM-TC significantly outperforms the traditional classifiers. It indicates that the text representations computed by the neural network can improve the performance of classification, and simply averaging the word embeddings cannot get good text representations.

Training traditional classifiers with weakly-labeled data. We further evaluate the performance of traditional classifiers with using both well-labeled and weakly-labeled data. We simply combine the 1000 well-labeled and all the weakly-labeled data as a whole training dataset to train the traditional classifiers. Table 5 compares the performance of traditional classifiers and LSTM-TC* in terms of accuracy and F1. It shows that even combining all the weakly-labeled data to train the baselines, the accuracies and F1 scores are still lower than the LSTM-TC*.

Table 5 Experimental results of training traditional classifiers with weakly-labeled data

	Accuracy	F1
LR (bigrams)	0.6364	0.6478
SVM (bigrams)	0.6891	0.6834
NB (bigrams)	0.6935	0.6977
XBGoost (bigrams)	0.6342	0.6989
LSTM-TC*	0.7808	0.7789

Evaluation on an unbalanced dataset. We further evaluate LSTM-TC and LSTM-TC* on an unbalanced dataset. The ratio of positive and negative samples in the well-labeled dataset and testing dataset is 1:5. For weakly-labeled dataset, the number of weakly-labeled positive and weakly-labeled negative samples is equal. We use AUC instead of accuracy to evaluate the performance. Figure 8 shows the ROC curve of LSTM-TC and LSTM-TC* on the unbalanced dataset. We can observe that LSTM-TC* has higher AUC than LSTM-TC, which indicates using weakly-labeled data can improve the classification performance on the unbalanced dataset.

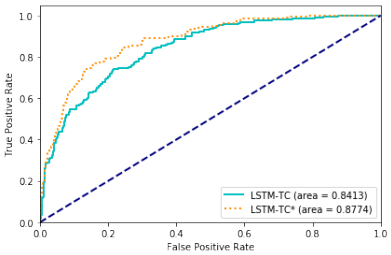

Fig. 8 ROC curve of LSTM-TC and LSTM-TC* on an unbalanced dataset

Effect of γ in Equation 9. γ is a scaling parameter that controls the scale of loss in Equation 9. We further evaluate how γ affects the classification performance (shown in Figure 9). Overall, the performance of LSTM-

TC* keep steady while changing the γ from 1 to 20. Hence, the performance of LSTM-TC* is not sensitive to the scaling parameter.

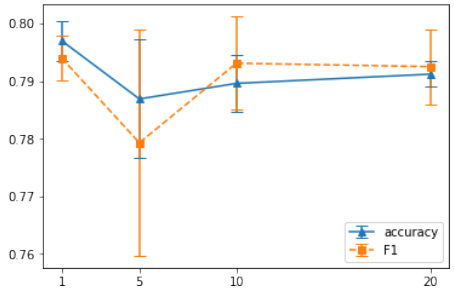

Fig. 9 Effect of γ for binary sentiment analysis

3.3.2 Multiclass Sentiment Analysis

Dataset. We adopt the Stanford Sentiment Treebank (SST) dataset with five classes (very positive, positive, neutral, negative, very negative). Similar to the binary sentiment analysis task, we adopt a subset of training data as golden training dataset T and use the rest of training data as the weakly-labeled dataset $\tilde{T}$. Each class of weakly-labeled data contains 80% of correct labeled data and 20% incorrect labeled data.

Results. We compare LSTM-Multi-TC* with baselines for multiclass sentiment analysis. In this experiment, we evaluate two various LSTM-Multi-TC* models. One adopts the mean operation (Equation 5) to compose the representation of each class, called *LSTM-Multi-TC* (mean)*. The other adopts the neural attention model (Equations 12-14) to compose the representation of each class, called *LSTM-Multi-TC* (attention)*. Table 6 shows accuracies and F1 scores of our proposed models and baselines with various sizes of golden training datasets. The overall results are similar to the results of binary sentiment analysis. The traditional classifiers have poor performance, which indicates the traditional classifiers cannot be well trained with limited well-labeled data. LSTM-Multi-TC* (attention) achieves better performance than the basic LSTM-Multi-TC, which shows the incorporation of weakly-labeled data in the pre-training phase of LSTM with the neural attention model for multiclass classification does improve the prediction accuracy. Meanwhile, compared with LSTM-Multi-TC* (mean), LSTM-Multi-TC* (attention) has higher accuracies and F1 scores on different sizes of golden training datasets. This is because, for a multiclass sentiment analysis task, a weakly-labeled "very positive sample" should be closer to positive samples than negative samples. Given a weakly-labeled "very positive" sample, the attention model can assign high weights to positive samples and low weights to negative samples. However, using the mean operation, all the positive and negative samples have the same weights.

4 Related Work

Deep neural networks have achieved promising results in sentence modeling and understanding for the purpose of classification, like sentiment analysis. The fundamental of applying deep neural networks for sentence classification is word embeddings (Bengio et al, 2003; Mikolov et al, 2013) and semantic composition (Blacoe and Lapata, 2012; Mitchell and Lapata, 2010; Turney, 2014; Zhu and Grefenstette, 2017).

Word embeddings map each word to a real-valued semantic vector. These word vectors are usually trained in an unsupervised way on a large text corpus. There are two types of approaches to train the word embeddings. The first one is based on neural probabilistic language model (Bengio et al, 2003). For example, Mikolov et al. (Mikolov et al, 2013) proposed the skip-gram model and continuous bag-of-words (CBOW) model. The skip-gram model trains the word embeddings by predicting the context words given the center word, while the CBOW model predicts the center word given the surrounding words. The second type is based on matrix factorization. For example, Lebret et al. (Lebret and Collobert, 2014) proposed a Helinger PCA to learn word representations by projecting the original word co-occurrence space to a dense subspace. Glove (Pennington et al, 2014) developed a method of finding the linear substructures of word co-occurrence matrix by a log-bilinear regression model. If the word embeddings are further fine-tuned based on a supervised training task, the vectors can encode the specific task information (Tang et al, 2014). Once the word vectors are well-trained, they can capture the hidden semantic and grammatical features of words. Thus, word embeddings can improve the accuracy of natural language processing tasks (Collobert et al, 2011).

Semantic composition constructs complex representations of phases or sentences by combining word embeddings. Various composition functions have been proposed. The basic model is based on algebraic operations, like additive or multiplication, to build sentence vector from word vectors (Mitchell and Lapata, 2010). Recently, using deep neural networks to compose the word embeddings to sentence embeddings has received increased attention. There are different kinds of neural networks for modeling sentence, including recursive neural networks (Socher et al, 2013), convolutional neural networks (Denil et al, 2014; Kim, 2014), recurrent

Table 6 Experimental results on multiclass sentiment analysis with various sizes of golden training datasets

Method	Accuracy				F1			
	Number of well-labeled data				Number of well-labeled data			
	200	500	800	1000	200	500	800	1000
LR (unigrams)	0.2258	0.2515	0.2484	0.2402	0.1257	0.1670	0.1739	0.1715
LR (bigrams)	0.2289	0.2402	0.2502	0.2484	0.1072	0.1335	0.1520	0.1541
SVM (unigrams)	0.2239	0.2240	0.2240	0.2366	0.1420	0.1422	0.1422	0.1737
SVM (bigrams)	0.2266	0.2425	0.2524	0.2447	0.1194	0.1467	0.1658	0.1610
NB (unigrams)	0.2171	0.2411	0.2389	0.2411	0.1287	0.1496	0.1549	0.1656
NB (bigrams)	0.2190	0.2425	0.2393	0.2434	0.1321	0.1560	0.1593	0.1768
XGBoost (unigrams)	0.2203	0.2402	0.2393	0.2402	0.1144	0.1379	0.1256	0.1302
XGBoost (bigrams)	0.2203	0.2407	0.2398	0.2396	0.1144	0.1391	0.1264	0.1290
CNN	0.2560 ± 0.0243	0.2933 ± 0.0105	0.3049 ± 0.0065	0.3041 ± 0.0082	0.1949 ± 0.0221	0.2365 ± 0.01470	0.2776 ± 0.0040	0.2903 ± 0.0097
LSTM-Multi-TC	0.3594 ± 0.0099	0.3781 ± 0.0049	0.3866 ± 0.0033	0.3860 ± 0.0053	0.3387 ± 0.0060	0.3669 ± 0.0016	0.3674 ± 0.0042	0.3746 ± 0.0057
LSTM-Multi-TC* (mean)	**0.3897± 0.0154**	0.3751 ± 0.0022	0.3827 ± 0.0055	0.3851 ± 0.0078	**0.3462 ± 0.0099**	0.3680 ± 0.0020	0.3686 ± 0.0143	0.3752 ± 0.0109
LSTM-Multi-TC* (attention)	0.3558 ± 0.0027	**0.3876± 0.0145**	**0.3926± 0.0177**	**0.3959± 0.0039**	0.3445 ± 0.0024	**0.3749± 0.0072**	**0.3741± 0.0261**	**0.3870± 0.0065**

neural networks (Graves, 2013; Tai et al, 2015). Mapping sentence to a semantic space based on deep neural networks for classification tasks usually achieves better results than the traditional approaches of representing a sentence as a combination of hand-designed features such as words or N-grams based on frequency.

The performance of semantic composition can be improved by using the attention mechanism. Attention mechanism is first proposed in sequence to sequence models for machine translation (Vaswani et al, 2017). With the attention mechanism, it predicts the target token by choosing the important source tokens automatically at each step. Some works adopt the attention mechanisms for semantic composition and further aim to document classification (Yang et al, 2016), document summarization (Paulus et al, 2018) and machine reading (Shen et al, 2017). In these works, the attention mechanisms are proposed to choose the important words for semantic composition.

The use of unlabeled data in deep learning has been investigated. The approach developed in (Hinton et al, 2006; Vincent et al, 2010) was to first use unlabeled data to pre-train the model layer by layer. As a result, the original data were projected into a low dimensional space. It then used labeled data to fine-tune the whole model. In (Weston et al, 2012), the authors proposed semi-supervised learning methods by adding a semi-supervised regularizer to deep neural networks. In (Socher et al, 2011), the authors used unlabeled data to pre-train the recursive neural networks for sentence modeling. However, one significant drawback of using unlabeled data for supervised tasks training is that they do not contain class label information. Different from all existing methods, our work introduces the first use of weakly-labeled data to pre-train the deep neural network for classification. We would like to emphasize that although our basic LSTM-TC and LSTM-Multi-TC models are not novel (as LSTM and word embeddings have been studied for text classification), the idea of using weakly-labeled data to pre-train the deep neural network is novel and the main contribution of this paper.

Discrimination generally refers to an unjustified distinction of individuals based on gender, race, or religion, and often occurs when the protected group (e.g., female) is treated less favorably than others. Discrimination discovery and prevention from historical datasets has been an active research area recently (Bonchi et al, 2015; Feldman et al, 2015; Hajian and Domingo-Ferrer, 2013; Pedreschi et al, 2013; Romei and Ruggieri, 2014). In this paper, we conducted the first study on identifying discrimination-related tweets from social networks. Our approach expects to be generally applicable for identifying tweets that satisfy some user-specified property, i.e., whether tweets contain private information.

5 Conclusions and Future Work

In this paper, we focus on tweet classification by using deep neural networks. We have developed a basic model LSTM-TC (LSTM-Multi-TC) for tweets binary (multiclass) classification. To improve the prediction accuracy, we have further developed an improved model LSTM-TC* (LSTM-Multi-TC*) that incorporates weakly-labeled data to pre-train the LSTM when lacking of a large training dataset.

Experimental results showed that comparing with traditional classifiers based on the hand-design features and existing deep neural networks without pre-training, our proposed pre-training model can significantly improve the prediction accuracy. We also evaluated how the quality and quantity of weakly-labeled data can affect the prediction accuracy. Our LSTM-TC* (LSTM-Multi-TC*) model can also be extended to other classification tasks such as sentiment analysis where large well-labeled training data are rarely available. Moreover, the pre-training idea using weakly labeled data can be portable to other deep neural networks (e.g., CNN) for training the representations of sentences. It expects to significantly improve the prediction accuracy of deep neural networks as semantic representations of words and sentences are the key ingredient of deep neural networks.

In the future, we plan to expand our work to the following directions. First, we will build a large corpus for tweet classification and conduct comprehensive experiments to examine the effectiveness and efficiency of the LSTM-TC* (LSTM-Multi-TC*) model. We will examine the use of other advanced deep learning models, e.g., Tree-LSTM (Tai et al, 2015), to determine whether better performance can be achieved. However, advanced models usually need a substantial amount of data. Finally, we are also interested to discover task-related sentences or words from long documents by using the attention-based deep neural networks (Rush et al, 2015).

Acknowledgments

The work is significant extended from our previous work (Yuan et al, 2016). In our previous work (Yuan et al, 2016), we proposed LSTM-TC and LSTM-TC* to deal with the binary classification task. In this work, we further extend our framework to address the multi-class classification task. The authors acknowledge the support from National Science Foundation (1646654) to Xintao Wu, and the National Natural Science Foundation of China (71571136) and the Research Project of Science and Technology Commission of Shanghai Municipality (14511108002,16JC1403000) to Yang Xiang.

References

Bahdanau D, Cho K, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: ICLR

Bamman D, Smith NA (2015) Contextualized sarcasm detection on twitter. In: ICWSM

Bastien F, Lamblin P, Pascanu R, Bergstra J, Goodfellow IJ, Bergeron A, Bouchard N, Bengio Y (2012) Theano: new features and speed improvements

Bengio Y, Simard P, Frasconi P (1997) Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on 5(2):157–166, DOI 10.1109/72.279181

Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. Journal of Machine Learning Research 3:1137–1155

Bengio Y, Courville A, Vincent P (2013) Representation learning: A review and new perspectives. TPAMI 35(8):1798–1828

Blacoe W, Lapata M (2012) A comparison of vector-based representations for semantic composition. In: EMNLP

Bonchi F, Hajian S, Mishra B, Ramazzotti D (2015) Exposing the probabilistic causal structure of discrimination. CoRR

Chen T, Guestrin C (2016) Xgboost: A scalable tree boosting system. In: Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, New York, NY, USA, KDD '16, pp 785–794, DOI 10.1145/2939672.2939785, URL http://doi.acm.org/10.1145/2939672.2939785

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

Denil M, Demiraj A, Kalchbrenner N, Blunsom P, de Freitas N (2014) Modelling, visualising and summarising documents with a single convolutional neural network. arXiv:14063830 [cs, stat]

Feldman M, Friedler SA, Moeller J, andSuresh Venkatasubramanian CS (2015) Certifying and removing disparate impact. In: KDD

Gambhir M, Gupta V (2017) Recent automatic text summarization techniques: a survey. Artificial Intelligence Review 47(1):1–66

Gers F, Schmidhuber J, Cummins F (1999) Learning to forget: continual prediction with lstm. In: ICANN

Graves A (2013) Generating sequences with recurrent neural networks. arXiv:13080850 [cs]

Hajian S, Domingo-Ferrer J (2013) A methodology for direct and indirect discrimination prevention in data mining. TKDE 25(7):1445–1459

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Computation 18(7):1527–1554

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

Irsoy O, Cardie C (2014) Opinion mining with deep recurrent neural networks. In: CoNLL

Iyyer M, Enns P, Boyd-Graber J, Resnik P (2014) Political ideology detection using recursive neural networks. In: ACL

Joshi A, Bhattacharyya P, Carman MJ (2017) Automatic sarcasm detection: A survey. ACM Comput Surv 50(5):73:1–73:22

Kim Y (2014) Convolutional neural networks for sentence classification. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pp 1746–1751

Lebret R, Collobert R (2014) Word embeddings through hellinger pca. In: EACL

LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444

LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. In: Neural networks: Tricks of the trade, Springer, pp 9–48

Mikolov T, Corrado G, Chen K, Dean J (2013) Efficient estimation of word representations in vector space. In: ICLR

Mitchell J, Lapata M (2010) Composition in distributional models of semantics. Cognitive Science 34(8):1388–1429

Paulus R, Xiong C, Socher R (2018) A deep reinforced model for abstractive summarization. ICLR

Pedreschi D, Ruggieri S, Turini F (2013) The discovery of discrimination. In: Discrimination and Privacy in the Information Society, pp 91–108

Pennington J, Socher R, Manning CD (2014) Glove: global vectors for word representation. In: EMNLP, pp 1532–1543

Rajadesingan A, Zafarani R, Liu H (2015) Sarcasm detection on twitter: A behavioral modeling approach. In: WSDM

Romei A, Ruggieri S (2014) A multidisciplinary survey on discrimination analysis. The Knowledge Engineering Review 29(05):582–638

Rush AM, Chopra S, Weston J (2015) A neural attention model for abstractive sentence summarization. arXiv:150900685 [cs]

Shen Y, Jin R, Chen J, He X, Gao J, Deng L (2015) A deep embedding model for co-occurrence learning. arXiv:150402824 [cs]

Shen Y, Huang PS, Gao J, Chen W (2017) Reasonet: Learning to stop reading in machine comprehension. In: KDD, ACM, KDD '17, pp 1047–1055

Socher R, Pennington J, Huang EH, Ng AY, Manning CD (2011) Semi-supervised recursive autoencoders for predicting sentiment distributions. In: EMNLP

Socher R, Huval B, Manning CD, Ng AY (2012) Semantic compositionality through recursive matrix-vector spaces. In: EMNLP

Socher R, Perelygin A, Wu JY, Chuang J, Manning CD, Ng AY, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp 1631–1642

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

Tai KS, Socher R, Manning CD (2015) Improved semantic representations from tree-structured long short-term memory networks. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pp 1556–1566

Tang D, Wei F, Yang N, Zhou M, Liu T, Qin B (2014) Learning sentiment-specific word embedding for twitter sentiment classification. In: ACL

Tomas M, Karafiat M, Burget L, Cernocky JH, Khudanpur S (2010) Recurrent neural network based language model. In: INTERSPEECH

Turney PD (2014) Semantic composition and decomposition: From recognition to generation. arXiv:14057908 [cs]

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. NIPS

Vincent P, Larochelle H, Lajoie I (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11:3371–3408

Weston J, Ratle F, Mobahi H, Collobert R (2012) Deep learning via semi-supervised embedding. In: Montavon G, Orr GB, Müller KR (eds) Neural Networks: Tricks of the Trade, Springer Berlin Heidelberg, no. 7700 in Lecture Notes in Computer Science, pp 639–655

Yang Z, Yang D, Dyer C, He X, Smola A, Hovy E (2016) Hierarchical attention networks for document classification. In: NAACL, pp 1480–1489

Yih Wt, Toutanova K, Platt J, Meek C (2011) Learning discriminative projections for text similarity measures. In: CoNLL

Yuan S, Wu X, Xiang Y (2016) Incorporating pre-training in long short-term memory networks for tweets classification. In: ICDM

Zeiler MD (2012) Adadelta: An adaptive learning rate method. arXiv:12125701 [cs]

Zhang X, Zhao J, LeCun Y (2015) Character-level convolutional networks for text classification. In: NIPS

Zhu X, Grefenstette E (2017) Deep learning for semantic composition. ACL

Response letter to review comments