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

Shuhan Yuan*, Xintao Wu† and Yang Xiang*
* Tongji University, Shanghai, China, Email: {4e66,shxiangyang}@tongji.edu.cn
† University of Arkansas, Fayetteville, AR, USA Email: xintaowu@uark.edu

***Abstract*—The paper presents deep learning models for tweets binary 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 develop two models for tweets classification. The basic model, called LSTM-TC, takes word embeddings as input, uses the LSTM layer to derive semantic tweet representation, and applies logistic regression to predict 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. 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 some traditional classifier that does not need a large amount of well-labeled training data. Our LSTM-TC* model first learns tweet representation based on the weakly-labeled data, and then trains the logistic regression classifier based on the small amount of well-labeled data. Experimental results show that: (1) the proposed method can be successfully used for tweets classification and outperform existing state-of-the-art methods; (2) pre-training tweet representation, which utilizes weakly-labeled tweets, can significantly improve the accuracy of tweets classification.**

***Index Terms*—LSTM, Tweets classification, Pre-training, Deep learning**

I. Introduction

In recent years, social network sites such as Twitter become important and influential as users actively share their stories and express their emotions or opinions there. Tweets classification in analyzing user-generated content (UGC) in social networks has become a hot research topic in natural language processing area. Various types of classification tasks like sentiment analysis, sarcasm detection or political ideology detection have been investigated [1]–[3]. Recent work in natural language processing has shown that deep neural networks [4], [5] can learn meaningful representations (or features) of text and use them to achieve high prediction accuracy in applications like sentiment analysis [6]–[9]. 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 a tweet 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 tweets classification model, called LSTM-TC, is 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 model sequentially takes each word in a tweet as input, extracts its salient information, and embeds each word into a semantic vector. When it reaches the last word, the hidden layer of the LSTM network provides a semantic representation of the whole tweet. On top of the tweet representation, our LSTM-TC applies a logistic regression classifier to identify tweet discrimination. The basic LSTM-TC model, like other deep learning models, still requires a large amount of training data to achieve good performance. We further develop an improved model, called LSTM-TC*, that incorporates a large amount of weakly-labeled tweets that are either available or can be easily constructed. Our LSTM-TC* first learns the tweet representation based on the large amount of weakly-labeled data in the pre-training phase, and then trains the logistic regression classifier based on the small amount of well-labeled data in the second phase. In general, the tweets that are weakly-labeled as positive are more likely to be truly positive than those weakly-labeled negative tweets. The pre-training phase of LSTM-TC* expects to learn better tweet representation by utilizing the similarity between the weakly-labeled tweets and well-labeled tweets, and hence LSTM-TC* can achieve better performance.

We show two approaches to construct the weakly-labeled data for tweets 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 [3], [10]. However, those tweets are actually weakly-labeled positive because those hastags 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 some 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 those traditional classifiers cannot achieve high prediction accuracy, they rarely require a large amount of training data.

To evaluate our proposed models, we focus on one case study — identifying discrimination-related tweets. In fact, we study how to determine whether a tweet is talking about discrimination or not. Firstly, we label a small number of tweets as the golden dataset. 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* 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 convolutional 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*.

II. Tweets Classification

Using deep neural networks for text classification is usually formed by first training the abstract representation of text by non-linear transformations and then training the classifier based on text representation as input. In this paper, we use LSTM to model the tweet representation and apply logistic regression for prediction. We first revisit the LSTM in Section II-A and then present the basic LSTM-TC model in Section II-B. In Section II-C, we present our improved LSTM-TC* model that uses the weakly-labeled tweets to train LSTM for learning tweet representation and uses the well-labeled data to train the classifier and fine-tune the whole model.

A. Long Shot-Term Memory Revisited

Recurrent Neural Networks (RNNs) are a class of deep neural networks and have been used extensively in time sequence modeling [11], [12]. 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 [13]. Long Shot-Term Memory (LSTM) recurrent neural network [14] was proposed to model temporal sequences and capture their long-range dependencies more accurately than the standard RNNs .

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 [15] which computes the hidden state $\mathbf{h_t} \in \mathbb{R}^{d_h}$ by:

$$\begin{aligned}
\mathbf{\tilde{c}_t} &= \tanh(\mathbf{W_c x_t} + \mathbf{U_c h_{(t-1)}} + \mathbf{b_c}) \\
\mathbf{i_t} &= \sigma(\mathbf{W_i x_t} + \mathbf{U_i h_{(t-1)}} + \mathbf{b_i}) \\
\mathbf{f_t} &= \sigma(\mathbf{W_f x_t} + \mathbf{U_f h_{(t-1)}} + \mathbf{b_f}) \\
\mathbf{o_t} &= \sigma(\mathbf{W_o x_t} + \mathbf{U_o h_{(t-1)}} + \mathbf{b_o}) \\
\mathbf{c_t} &= \mathbf{i_t} \odot \mathbf{\tilde{c}_t} + \mathbf{f_t} \odot \mathbf{c_{t-1}} \\
\mathbf{h_t} &= \mathbf{o_t} \odot \tanh(\mathbf{c_t})
\end{aligned} \tag{1}$$

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 vectors; $\mathbf{h}$ is the hidden vector; $\mathbf{W}$ and $\mathbf{U}$ are the parameters of weight; $\mathbf{b}$ is the bias term. 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 networks, 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.

B. Basic LSTM-TC Model

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 [16]–[18]. 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 [19]–[21]. 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 input 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 $\mathbf{r} \in \mathbb{R}^{d_h}$:

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

Tweets 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:

$$P(\hat{y}_i|\mathbf{r_i}, \mathbf{U_l}, b_l) = \frac{1}{1 + e^{-(\mathbf{U_l}\cdot\mathbf{r_i}+b_l)}}, \tag{3}$$

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 model aims to minimize the cross entropy as loss function to optimize the model. The cross entropy is defined as:

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

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

Algorithm 1 shows the complete training method for tweets classification. We iterate our algorithm $Epoch$ times. In each running, we first compute tweet representation r_i (Line 4), fit the logistic model (Lines 5-6), and update θ (Line 7). The parameter θ is optimized by Adadelta using back-propagation algorithm. Adadelta [22] is an advanced approach for gradient decent, which provides a per-dimension learning rate. Adadelta can speed up the convergence rate comparing to the traditional stochastic gradient descent (SGD) approach. We use dropout [23] 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 [24] to accelerate the training procedure.

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

```
Inputs  : Training dataset $T$
          Maximum training epoch $Epoch$
Outputs: Trained Tweets Classification Model
1  $j \leftarrow 0$
2  while $j < Epoch$ do
3      for each tweet $i$ in $T$ do
4          compute $\mathbf{r_i}$ by (1,2) on $T$;
5          compute $\hat{y}_i$ by (3) based on $\mathbf{r_i}$;
6          compute $\boldsymbol{L}_1(\hat{y}_i;\theta)$ by (4);
7          update $\theta$ with Adadelta;
8      end for
9      $j \leftarrow j + 1$;
10 end while
```

C. Improved LSTM-TC* Model

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

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 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 input, 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:

$$\mathbf{q}^+ = \frac{1}{M}\sum_{i=1}^{M}\mathbf{r_i^+}, \quad (5)$$

where M is the number of tweets in T^+. Similarly we compute $\mathbf{q}^-$ 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 $\tilde{\mathbf{r}}_\mathbf{i}^+$ and negative representation $\mathbf{q}^+$:

$$sim(\tilde{\mathbf{r}}_\mathbf{i}^+, \mathbf{q}^+) = \frac{\tilde{\mathbf{r}}_\mathbf{i}^+ \cdot \mathbf{q}^+}{\|\tilde{\mathbf{r}}_\mathbf{i}^+\|\|\mathbf{q}^+\|}, \quad (6)$$

where $\|\cdot\|$ denotes the L2 norm. The similarity score between the weakly-labeled positive tweet representation $\tilde{\mathbf{r}}_\mathbf{i}^+$ and negative representation $\mathbf{q}^-$ is denoted as $sim(\tilde{\mathbf{r}}_\mathbf{i}^+, \mathbf{q}^-)$. Because the model computes the similarity pairwisely, we adopt the pairwise learning loss function [25] to train the model.

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

$$\delta = sim(\tilde{\mathbf{r}}_\mathbf{i}^+, \mathbf{q}^+) - sim(\tilde{\mathbf{r}}_\mathbf{i}^+, \mathbf{q}^-). \quad (7)$$

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

$$\delta = sim(\tilde{\mathbf{r}}_\mathbf{i}^-, \mathbf{q}^-) - sim(\tilde{\mathbf{r}}_\mathbf{i}^-, \mathbf{q}^+). \quad (8)$$

We use logistic loss function over δ to train the parameters:

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

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

Because the similarity function is 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 positive 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.

III. Experiments

In this work, we evaluate our method by identifying discrimination-related tweets. In particular we focus on identifying two types of discrimination: sexism and racism.

A. Setup

Dataset. Because there is no benchmark dataset for discrimination discovery, we built our dataset by crawling 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 [1] which provide instances of discrimination stories. To build the training dataset T, we manually label and include 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 built 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 classifer. 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 (non-)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 [2] 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.

Word Embeddings and Hyperparameters. Word embeddings are required as input in our framework. Our tweet dataset contains 17320 unique words. We use the off-the-shelf pre-trained word embeddings [3] provided by Mikolov [17]. These word embeddings are widely used and have been shown to achieve good performance on many NLP tasks. We find word embeddings of 13000 words there and use uniform distribution to initialize the remaining 200 words that appear at least 5 times. The dimension of each word embedding is

[1] http://everydaysexism.com/, http://stemfeminist.com/
[2] http://www.cs.cmu.edu/~ark/TweetNLP/
[3] https://code.google.com/archive/p/word2vec/

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 stop. The training epoch is 100 with early stop. 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 for discrimination detection, logistic regression (**LR**), support vector machine (**SVM**), naive Bayes classifier (**NB**), and convolutional neural network (**CNN**). The first three classifiers are trained on traditional N-gram features, in particular, unigram and bigram features. The **CNN** is trained on word embeddings. We follow the procedure proposed in [6] to build the convolutional neural network. We implement all deep neural networks using the Theano package [4] and train the SVM, naive Bayes, and logistic regression models using the Scikit-learn package [5].

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

B. 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.

Comparisons. Table I 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. Furthermore, compared with the CNN, our LSTM models have a clear trend, i.e., the increase of the training data improves the prediction accuracy. On the contrary, the CNN does not change much when we change the well-labeled data from 120 to 480.

Effect of the size of the weakly-labeled dataset. We use different sizes of the weakly-labeled dataset to 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 1 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.

[4]http://deeplearning.net/software/theano/
[5]http://scikit-learn.org/stable/

TABLE I
EXPERIMENTAL RESULTS ON DISCRIMINATION RELATED TWEETS DETECTION

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
CNN	0.892	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**

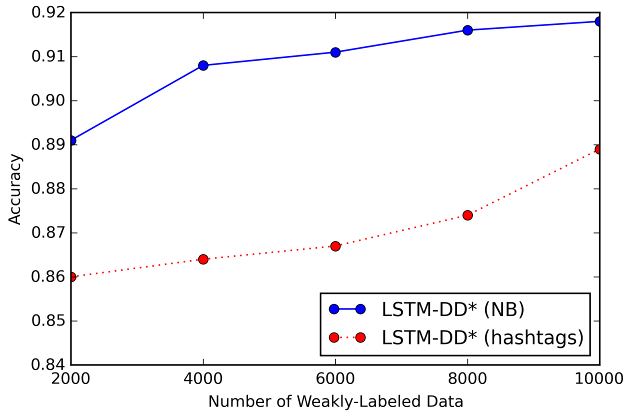

Fig. 1. Effect of the size of the weakly-labeled dataset

IV. RELATED WORK

Deep neural networks have achieved promising results in sentence modeling and understanding for purpose of classification, like sentiment analysis. The fundamental of applying deep neural networks for sentence classification is word embeddings [16], [17], [27] and semantic composition [9], [19], [21], [26]. There are different kinds of neural networks for modeling sentence, including recursive neural networks [9], convolutional neural networks [6], [28], recurrent neural networks [12], [29], and paragraph vector [30]. Mapping sentence to a semantic space based on deep neural networks 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 use of unlabeled data in deep learning has been investigated [31]–[34]. Different from all existing methods, our work introduces the use of weakly-labeled data to train the deep neural network.

V. Conclusions and Future Work

In this paper, we have developed a basic model LSTM-TC for tweets classification. To improve the prediction accuracy, we further developed an improved model LSTM-TC* that incorporates weakly-labeled data to pre-train the LSTM when lacking of large training dataset. We evaluated LSTM-TC* with two types of weakly-labeled datasets, based on hashtag information and naive Bayes classifier output respectively. Experimental results showed that comparing with traditional classifiers based on the traditional hand-design features, the proposed models can significantly improve the prediction performance. We also evaluated how the quality and quantity of weakly-labeled data can affect the prediction accuracy.

In our future work, we plan to build a large corpus for tweet discrimination detection and conduct comprehensive experiments to examine the effectiveness and efficiency of the LSTM-TC* model. We also plan to study discrimination detection in multiple categories, e.g., predicting a tweet in a specific category such as sexism, racism and ageism.

Acknowledgments

The authors acknowledge the support from the 973 Program of China (2014CB340404), the National Natural Science Foundation of China (71571136), the Basic Research Program of Shanghai (16JC1403000), and China Scholarship Council to Shuhan Yuan and Yang Xiang, and from National Science Foundation (1646654) to Xintao Wu. This research was conducted while Shuhan Yuan visited University of Arkansas. Yang Xiang is the corresponding author of the paper.

References

[1] D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin, "Learning sentiment-specific word embedding for twitter sentiment classification," in *ACL*, 2014.

[2] M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik, "Political ideology detection using recursive neural networks," in *ACL*, 2014.

[3] A. Rajadesingan, R. Zafarani, and H. Liu, "Sarcasm detection on twitter: A behavioral modeling approach," in *WSDM*, 2015.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *TPAMI*, vol. 35, no. 8, pp. 1798–1828, 2013.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*, 2014.

[7] O. Irsoy and C. Cardie, "Opinion mining with deep recurrent neural networks," in *CoNLL*, 2014.

[8] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *NIPS*, 2015.

[9] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *EMNLP*, 2013.

[10] D. Bamman and N. A. Smith, "Contextualized sarcasm detection on twitter," in *ICWSM*, 2015.

[11] M. Tomas, M. Karafiat, L. Burget, J. H. Cernocky, and S. Khudanpur, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

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

[13] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1997.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] F. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with lstm," in *ICANN 1999*, 1999.

[16] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[17] T. Mikolov, G. Corrado, K. Chen, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[18] Y. Shen, R. Jin, J. Chen, X. He, J. Gao, and L. Deng, "A deep embedding model for co-occurrence learning," *arXiv:1504.02824 [cs]*, 2015.

[19] J. Mitchell and M. Lapata, "Composition in distributional models of semantics," *Cognitive Science*, vol. 34, no. 8, pp. 1388–1429, 2010.

[20] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng, "Semantic compositionality through recursive matrix-vector spaces," in *EMNLP*, 2012.

[21] W. Blacoe and M. Lapata, "A comparison of vector-based representations for semantic composition," in *EMNLP*, 2012.

[22] M. D. Zeiler, "Adadelta: An adaptive learning rate method," *arXiv:1212.5701 [cs]*, 2012.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[24] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Mller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[25] W.-t. Yih, K. Toutanova, J. Platt, and C. Meek, "Learning discriminative projections for text similarity measures," in *CoNLL*, 2011.

[26] P. D. Turney, "Semantic composition and decomposition: From recognition to generation," *arXiv:1405.7908 [cs]*, 2014.

[27] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *Journal of Machine Learning Research*, no. 12, pp. 2493–2537, 2011.

[28] M. Denil, A. Demiraj, N. Kalchbrenner, P. Blunsom, and N. de Freitas, "Modelling, visualising and summarising documents with a single convolutional neural network," *arXiv:1406.3830 [cs, stat]*, 2014.

[29] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *ACL*, 2015.

[30] L. Quoc and M. Tomas, "Distributed representations of sentences and documents," in *ICML*, 2014.

[31] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," 2006.

[32] P. Vincent, H. Larochelle, and I. Lajoie, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[33] J. Weston, F. Ratle, H. Mobahi, and R. Collobert, "Deep learning via semi-supervised embedding," in *Neural Networks: Tricks of the Trade*, ser. Lecture Notes in Computer Science, G. Montavon, G. B. Orr, and K.-R. Mller, Eds., no. 7700. Springer Berlin Heidelberg, 2012, pp. 639–655.

[34] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, "Semi-supervised recursive autoencoders for predicting sentiment distributions," in *EMNLP*, 2011.