Are Deep Learning Methods Better for Twitter Sentiment Analysis?

Yujie Lu Kotaro Sakamoto Hideyuki Shibuki Tatsunori Mori
Graduate School of Environment and Information Sciences, Yokohama National University
{luyujie, sakamoto, shib, mori}@forest.eis.ynu.ac.jp

1 Introduction

Many applications based on sentiment analysis on social media, such as Twitter, have been developed by researchers. Recently, during the Unites States presidential election of 2016, politicians, including president-elect Donald J. Trump, have been using Twitter as a mean of communicating with the public, drawing tremendous attention to Twitter[1].

It is known that sentiment analysis on tweets still fall behind it on traditional texts, such as movie reviews and product reviews, while it has achieved tangible progress since the popularity of deep learning and the availability of large training datasets (distant supervised by emoticons and hashtags). However, previous sentiment analysis studies on tweets usually carried out in an experimental setting, i.e., their training/validation/test datasets only have two types of global polarities (i.e., positive and negative). What makes it worse, the judgments made by systems are not consciously linked to the designated evaluation objects, i.e., they just mathematically fit features extracted from tweets to their true polarities as simple document classification does.

These experimental settings are basically unpractical for real-world systems. First, the tweets that are passed to systems in a production environment unavoidably contain three types of polarities (i.e., positive, negative and neutral), since we cannot pick out positive and negative tweets from a large collection of tweets by any kind of automated tweet filtering beforehand.

Second, the global polarity of a tweet is defined as author's attitude to a specific evaluation object (topic) in it. We cannot decide the global polarity of a tweet without given an evaluation object, especially for tweets containing multiple topics and complex contexts. For example, in the following tweet, there are two topics (i.e., #windows8 and #skype). This tweet could be a positive tweet (regarding Windows 8) or a neutral tweet (regarding Skype).

So now on #windows8, any time #skype plays a sound to my speakers, it breaks all speaker sound for everything, even across reboots. Lovely.

From the above, we can see that Twitter sentiment analysis in a real-world setting is more difficult than it is in an experimental setting[2]. [3] constructed an unbiasedly-selected and carefully-annotated multilingual tweet corpus (called as the MDSU corpus below) for sentiment analysis[3]. The MDSU corpus is not only close to the real-world setting (the global polarity of each tweet is one of the three types and topic-relevant), but also is full of complex contexts.

In this paper, we will use its English dataset to test three different models (i.e., SVM[4], CNN[5], and RNN[6]) to see how well they perform in the a real-world setting and whether can the deep learning methods always outperform the SVM baselines.

2 Related Work

In this section, we briefly summarize the previous studies on Twitter sentiment analysis.

2.1 Machine Learning Methods

As an early attempt, [1] annotated a noisy-labeled tweet dataset by emoticons, carried out experiments with three machine learning methods, including SVM, Maximum Entropy, and Naive Bayes, fed with binary unigram/ bigram features, same as [4] conducted on movie reviews, and showed that the SVM model achieved the best accuracy. Later on, many researchers tried various machine learning methods fed with their own specially-designed features, such as n-gram, POS, synonym, topic, word

[1]Please refer to the article from the New York Times: For Election Day Influence, Twitter Ruled Social Media

[2]In this paper, the experimental setting regards Twitter sentiment analysis as a binary classification task without specified evaluation objects, while the real-world setting regards it as a 3-class classification task with specified evaluation objects

[3]Although the MDSU corpus contains rich annotations, we mainly use it as a testbed in the paper.

[4]SVM stands for Support Vector Machine.

[5]CNN stands for Convolutional Neural Network.

[6]RNN stands for Recurrent Neural Network.

polarity, emoticon, context, discourse relation, and even, author gender. The SemEval-2014 Task 9 report also pointed out that most participants resorted to machine learning methods with various features, who depended heavily on sentiment lexicons, attaining approximately 70% accuracy at best.

2.2 Deep Learning Methods

Due to the prevalence of deep learning in these years, many different network structures have been put forth for opinion mining. [2, 6] introduced their attempts of using CNN and recursive neural networks on sentence-level sentiment classification respectively, achieving rather inspiring results. Later on, researchers started to apply them to Twitter sentiment analysis. [5] explored the deep convolutional neural networks, and achieved accuracies that could rank in the first two positions in Semeval-2015 Task 10. [7] proposed a Long Short Term Memory (LSTM) recurrent network, and outperformed several feature-engineering approaches. They also reported that their models are able to capture the special functions of words. Besides these studies, a couple of other deep network structures have been proposed.

Although our models to be used for comparison do not have significant differences from the existing methods, it is still meaningful to observe their adaptability and performance in the real-world setting.

3 Models

3.1 Support Vector Machine

SVM has proved to be an efficient classification model for document classification. The essence of SVM is to find a hyperplane represented by its normal vector $\mathbf{w}$ which maximizes the margin between two classes. This search then becomes a constrained optimization problem (specifically speaking, a solved convex quadratic programming problem) and the solution can be written as:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i x_i, \quad \alpha_i \geq 0 \tag{1}$$

where x_i are support vectors lying on the class boundaries, α_i are coefficients of support vectors and y_i are true values, each of which $\in \{1, -1\}$.

To solve multiclass classification task, one-vs-the-rest or one-vs-one strategy can be adopted. By default, the one-vs-one strategy is used. Further, for linearly inseparable problems, kernel trick can be used.

The machine learning methods based on SVM with n-gram features proposed by Pang et al. (2002) and Go et al. (2009) are frequently used as baselines in many previous studies. Similar to their settings, we use the default SVM model with a linear kernel and $C = 1$. Besides, the binarized unigram/bigram term frequencies are used as our features. The models are trained with LibSVM (Chang and Lin, 2011) via Python scikit-learn library.

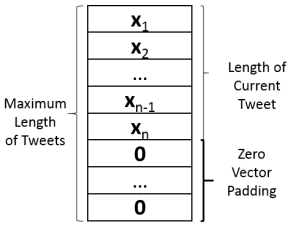

Figure 1: Tweet Matrix with Zero Padding

3.2 Convolutional Neutral Network

One of the advantages of CNNs is that they have many fewer parameters than fully connected networks with the same number of hidden units, which makes them much easier to be trained. Our CNN architecture is very close to the architecture of [2].

Let $\mathbf{x}_i$ be the k-dimensional word vector corresponding to the i-th word in a tweet; thus, a tweet having n words is represented as:

$$\mathbf{x}_{1:n} = \mathbf{x}_1 \oplus \mathbf{x}_2 \oplus \mathbf{x}_3 \oplus ... \oplus \mathbf{x}_n \tag{2}$$

where is $\oplus$ the concatenation operator. To unify the matrix representation of tweets in different length, the maximum length of all tweets in the dataset is used as the fixed length for tweet matrices. For shorter tweets, zero vector was padded at the back of a tweet matrix (see Figure 1).

Next, we carried out convolution operation across each tweet matrix to transform it to a scalar c_i. This procedure is formulated as follow.

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \tag{3}$$

$$\mathbf{c} = [c_1, c_2, ..., c_{n-h+1}] \tag{4}$$

where $\mathbf{w}$ denotes a filter map, h is the window size of a filter, f is a non-linear activation function and b is a bias term. By doing so, reginal word vectors $\mathbf{x}_{i:i+h-1}$ in a tweet matrix is convoluted to c_i.

Then, we conduct subsampling operation. Here, we use the following max-pooling as our subsampling method.

$$c_{max} = max\{\mathbf{c}\} \tag{5}$$

From Eqs. (3)–(5), a filter generates one c_{max} from a tweet matrix. Practically, the convolution and subsampling operations are usually performed in pair.

The number of filter maps of our CNN model is 100, and the possible window sizes are $\{3, 4, 5\}$, so there are 300 different filters in total in our model.

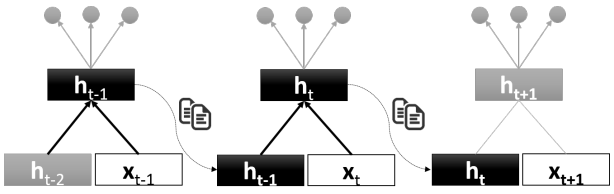

Figure 2: Illustraion of Our RNN Model

The corresponding 300 c_{max} form the penultimate layer, and are then passed to a fully connected softmax layer to predict the global polarity of a tweet.

Further, we employed Dropout as our regularization on the penultimate layer. Dropout was proved to be an effective way to prevent co-adaptation of hidden units by randomly setting a portion of the hidden units to zeroes during feedforward/backpropagation.

3.3 Recurrent Neutral Network

RNNs have gained tremendous attention in the NLP field, and they have been employed to handle many tasks, including machine translation. One of the characteristics of RNNs is that they can use their internal memory to process sequences of inputs in arbitrary length.

A RNN consists of a hidden state $\mathbf{h}$ and an optional output $\mathbf{y}$ which operates on a variable-length sequence $\mathbf{x} = (x_1, ..., x_T)$. At each time step t, the hidden state $\mathbf{h}_{\langle t \rangle}$ of the RNN is updated by

$$\mathbf{h}_{\langle t \rangle} = f(\mathbf{h}_{\langle t-1 \rangle}, x_t) \tag{6}$$

where f is a non-linear activation function. Unlike other neutral networks, researchers have designed quite complex activation functions for RNNs, such as LSTM and GRU (Gated Recurrent Unit) cells.

After the last word vector being inputed to the model, the polarity distribution of the global polarity of a tweet can be given by the softmax layer using $\mathbf{h}_{\langle T \rangle}$, as follows.

$$p(\mathbf{y}_j = 1) = \frac{exp(\mathbf{w}_j \mathbf{h}_{\langle T \rangle})}{\sum_{j'}^{K} exp(\mathbf{w}_{j'} \mathbf{h}_{\langle T \rangle})} \tag{7}$$

where K is the number of classes, $j = 0, ..., K-1$, and $\mathbf{w}_j$ are the rows of the weight matrix $\mathbf{W}$ of the softmax layer.

By reusing the hidden units in the previous layer, RNNs allow past information sequentially encoded inside the networks. Such structures make it possible to compress a variable-length input into a fix-length vector $\mathbf{h}_{\langle T \rangle}$. Figure 2 illustrates our RNN model. Our RNN architecture is basically as same as the Elman Network described in [7]. The size of the hidden layer is 100 in this paper.

4 Experiment

4.1 Experiment Setup

The MDSU corpus involves 3 languages (i.e., English, Japanese and Chinese) and 4 international topics (i.e., iPhone 6, Windows 8, Vladimir Putin, and Scottish Independence), consisting of 12 collections. Totally, the corpus has 5422 tweets, with each collection containing approximately 450 tweets. The English and Japanese tweets are collected from Twitter[7], and the Chinese tweets are collected from Weibo[8], a Chinese version of Twitter. During the tweet selection, we filtered out those apparent non-emotional tweets and preferred to choose tweets with rich language phenomenon.

As said in the introduction section, we only use the English dataset of the MDSU corpus, whose polarity distribution is shown in Table 1. The average number of words of the tweets in the dataset is 23, with a minimum of 10 and a maximum of 34 (i.e., the length of tweet matrices for the CNN model). The instances of the dataset is given in the [<topic>, <tweet text>, <polarity>] format, such as ["Windows 8", the example tweet in Section 1, "positive"].

Table 1: Polarity Distribution of the English Dataset

Positive #	Negative #	Neutral #
503	774	526

In this paper, the word embedding we used for our deep learning modes is `word2vec`[9] vectors trained on Google News, who have dimensionality of 300. Words not present in `word2vec` vectors are initialized randomly (using a uniform distribution having approximately the same variance with the pre-trained vectors). Our pre-trained vectors are treated as fixed inputs, and are not fine-tuned during learning.

There are two types of classification tasks in this paper. One is a 3-class classification (all 1803 instances in [<topic>, <tweet text>, <polarity>] format), and the other is a binary classification (1277 non-neutral instances in [<tweet text>, <polarity>] format). For all the models, we use accuracy as our metric, and 10-fold cross validation for model evaluation. For each fold, CNN models run 25 epochs and RNN models run 100 epochs. To simplify the comparison, we only used word information as inputs for all three models (binarized unigram/bigram frequencies for SVM models, and word vectors for CNN and RNN models).

[7] http://www.twitter.com
[8] http://weibo.com
[9] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

Table 2: Classification Accuracy of Each Model

Type	Model	Accuracy
Binary Classification	SVM (unigram)	0.746
	SVM (bigram)	0.714
	SVM (unigram+bigram)	**0.784**
	CNN (max-pooling)	0.676
	RNN (Elman)	0.587
3-Class Classification	SVM (unigram)	0.549
	SVM (bigram)	0.500
	SVM (unigram+bigram)	**0.588**
	CNN (max-pooling)	0.504
	RNN (Elman)	0.417

4.2 Results and Discussion

Table 2 shows the accuracy of each model for both tasks. From Table 2, we find that the SVM model using both unigram and bigram features obtains the best results for both binary and 3-class classification in all five models. For binary classification, it attains the 78.4% accuracy, and for 3-class classification, it attains 58.8% accuracy. We also tested the SVM models using longer n-grams ($n \geq 3$), and it showed no improvement.

The two deep learning models underperform most of the SVM models. For binary and 3-class classification, CNN model attains 67.6% and 50.4% (only better than the SVM model using bigram) accuracies respectively, and the RNN model only gains 58.7% and 41.7% accuracies respectively. We speculate that these mediocre results are mainly owing to the small size of our training data relative to model complexity of the two models (i.e., the parameter number is on the order of tens of thousands for the RNN model, and hundreds of thousands for the CNN model). We observed the performance of both models on training data, and found that the CNN model obtain 99% accuracy after 15 epochs, and the RNN model does so after approximately 130 epochs. This means both the CNN and RNN models had very bad generalization (i.e., overfitting) ability and lacked robustness.

Further, the SVM models with unigram/bigram feature were also experimented by Go and Pang on their datasets (i.e., noisy-labelled tweets and movie reviews). We can see that the performance of the same model on the MDSU corpus (i.e., 74.6%/71.4%) deceased to a certain extent comparing with Go and Pang (i.e., 82.2%/81.6% and 82.9%/77.1%, respectively). The decrease may be caused by the complexity of our dataset. Although researchers reported 70%∼80% accuracy for Twitter sentiment analysis using their methods, from Table 2, we can see that the accuracies are much lower in the real-world setting (i.e., 3-class classification).

5 Conclusion and Future Work

By experimenting with different models on our MSDU corpus, we have three main conclusions. First, deep learning methods are not necessarily better than traditional machine learning methods. Their performance not only depends on the appropriateness of the network structure, but also on the size of the training dataset. Second, the classification accuracies in the real-world setting is much lower than them in the experimental setting, which means the current reported performance of Twitter sentiment analysis in previous studies may be over-claimed. Third, we reconfirmed that SVM models have rather good fitness to NLP tasks, especially when the size of dataset is limited. In the future, we will focus on the understanding of the special contexts in tweets, and try to put forth tree-based methods to model the intrinsic logic of sentiment contained in tweets.

References

[1] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report (Stanford)*, pages 1–6, 2009.

[2] Yoon Kim. Convolutional neural networks for sentence classification. *Proceedings of EMNLP 2014*, pages 1746–1751, 2014.

[3] Yujie Lu, Kotaro Sakamoto, Hideyuki Shibuki, and Tatsunori Mori. Construction of a multilingual annotated corpus for deep sentiment understanding in social media. *IPSJ SIG Technical Reports*, 2015-NL-222(1):1–12, 2015.

[4] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. *Proceedings of EMNLP 2002*, pages 79–86, 2002.

[5] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. *Proceedings of SIGIR'15*, pages 959–962, 2015.

[6] Richard Socher, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. *Proceedings of EMNLP 2013*, pages 1631–1642, 2013.

[7] Xin Wang, Yuanchao Liu, Chengjie Sun, Baoxun Wang, and Xiaolong Wang. Predicting polarities of tweets by composing word embeddings with long short-term memory. *Proceedings of ACL/IJCNLP 2015*, pages 1343–353, 2015.