Deep Tweeting: Twitter Sentiment Analysis using Recursive Deep Models

Nan Xia
Michigan State University
nanxia@msu.edu

Zhuangdi Zhu
Michigan State University
zhuzhuan@msu.edu

Abstract—**In this project, we explore the application of Recursive Neural Network (RNN) models on the sentiment analysis task with Twitter. Twitter is a dominating social media that allows users to post 140-character long micro blogs called tweets, which usually convey large amount of information that can be further used for opinion mining on important social events. Different from prior work using bag-of-words models, we plan to evaluate how deep learning models can be used to improve the accuracy of sentence-level sentiment analysis in micro-blogging domain. We use word-to-vector approaches to map each word in a tweet into a vector representation which capture the semantic relevance among different words, and use the vector representation as inputs to training our RNN classifier. Our RNN approach outperforms traditional bag-of-words based classifiers by over 20% percent accuracy. It also outperforms a Convolution Neural Network based approach by over % percent accuracy, with a faster convergence rate. Our model can also be scaled to solving multi-class classification problems.**

I. Introduction

A. Motivation

In this project, we will propose and implement deep learning models to conduct text sentiment analysis. Especially, we will use Recurrent Neural Network models to classify sentiment labels of tweets, and see how deep learning models can improve the classification accuracy, compared with bag-of-words methods such Naive Bayes or Support Vector Machine. Twitter is a popular microblogging platform that allow users to post 140-character-constrained microblogs called tweets, which reveal large amount of information about public opinion in real time [13]. The results of sentiment analysis on twitter can be further used for opinion mining in many socio - economical phenomena. Prior work has applied twitter sentiment analysis into areas such as president-election prediction [16], [17], market targetting [6], and stock price estimation [7]. It is a meaningful research area that will continue to attract interests and efforts from academia.

B. Background

The explosion of data about public opinion on social networks and review websites in recent years leads to an ever-growing demand in opinion mining in many soci-economical areas. This demand can be satisfied by sentiment analysis, a popular NLP technique that relies on large amount of data labeled with sentiment categories. With the easily achievable online data that reveal user sentiments, the potential of sentiment analysis has been exploited in various areas, such as high frequency trading, election prediction, and stock price estimation.

A basic method of sentiment analysis is to use bag-of-words based model. This kind of models ignore the order of words in a context, and the predicted label is dominated by a few words with strong sentiment. The most frequently studied model based on bag-of-word features is Naive Bayes, which is a probabilistic classifier. The rationale behind Naive Bayes model is that given a document, it will generate the class label that has the maximum posteriror probability given the document. This model was first applied to text classification by Mosteller and Wallace in 1964 [12]. Bag-of-words based models work well in document-level analysis. However, for short reviews or comments in one sentence, the classification accuracy of this model is around 80% [17]. Since the dominant type of data online is usually short comments or reviews, there is a need to better predict sentiment in a sentence-level.

Another kind of methods is to is to use deep learning approaches. With the blossom of large amount of corpus available on the internet, deep learning approaches using neural networks have earned increasing popularity the filed of Natural Language Processing. Researchers from Stanford University have proposed complex neural network models to conduct sentence-level sentiment analysis. An example is the Matrix-Vector Recursive Neural Network (MV-RNN), proposed by Socher *et al.* [14]. This kind of models takes input a parse tree of a sentence, and generates the sentiment category of the root node of the parse tree by recursively building it in a bottom-up fasion. They work well only for sentences follow correct gramma or syntactic rules, but are not suitable for tweets, which have informal expression both in gramma and in lexicon.

II. Related Work

There are several previous researches on NLP techniques designed for Twitter data. Hereby are some most relevant work for our reference:

Kouloumpis *et al.* conducted sentiment analysis on twitter data, and their focus is to evaluate the effects of different kinds of features on the classification accuracy [10]. They explored the features of n-gram, POS, and especially, micro-blog features such as emotions and hastags. Their findings are that POS features not be useful for twitter sentiment analysis, while micro-blog features are best indicators of sentiment labels. Their best performance is 0.75 on average accuracy.

Apoorv *et al.* proposed a tree-kernel based model to do sentiment analysis [5]. They designed a new tree representation for tweets as inputs to the kernel to obviate the need of extracting features. The best performance of their model is 60.83% accuracy.

Wang *et al.* use a naive Bayes model to conduct twitter sentiment analysis in order to analyze the president election cycle [17]. They adopted unigram features as the classifier input. The key novelty of this work is that they implemented an analysis system that takes real-time twitter messages as inputs, instead of doing post-facto analysis. Their discovery is that tweet volume is largely driven by campaign events.

Go *et al.* use distant supervision to conduct Twitter sentiment classification [8]. The key novelty of their work is the use of emotion features for distant supervised learning. They explored with different classifiers, such asNaive Bayes, Maximum Entropy, and SVM, with a best performance of around 80% for average accuracy.

More neural network-based sentiment analysis models have been proposed recently. Kim *et al.* used a standard Convolution Neural Networks to classify tweets, with a best performance of 89.6% Socher *et al.* created a recursive model to build the sentiment label of a sentence through its parse tree [14]. Each constituent in the sentence is associated with a vector-matrix representation.The vector conveys the meaning of the constituent, while the matrix conveys how it changes the meaning of neighboring constituents.

III. DeepTweeting Sentiment Analysis System

We implement a system that utilizes a recursive a Recurrent Neural Network (RNN) model to predict the sentiment classes of different tweets. Especially, we use a specialized RNN named Long Short Term Memory (LSTM) network as our system model. We adopt two pre-processing steps to map each tweet into a three-dimensional matrix representation, so that the relevant meaning among different words in the Twitter corpus can be captured. The output of our model is a probability distribution of all the sentiment labels for a tweet. In our project, we only consider two-class sentiment classifications, that is, a tweet can be either negative or positive. However, our model can be easily scaled to multi-sentiment classification problems. Our system overview is shown in Figure 1.

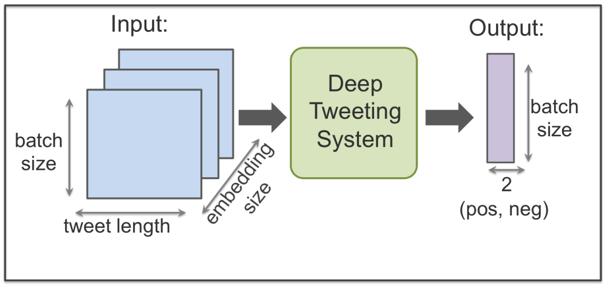


Fig. 1: Sentiment Analysis System Overview

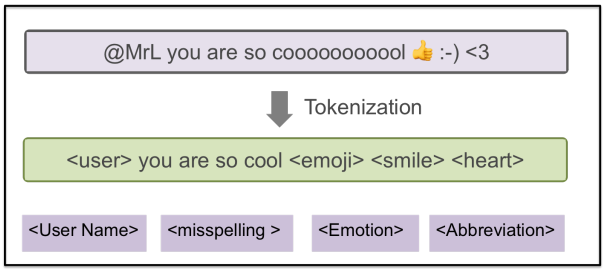
A. Preprocessing

We use word embedding to map Chinese characters from the poem corpus to vectors of real numbers, which consists of two steps: tokenization and word embedding.

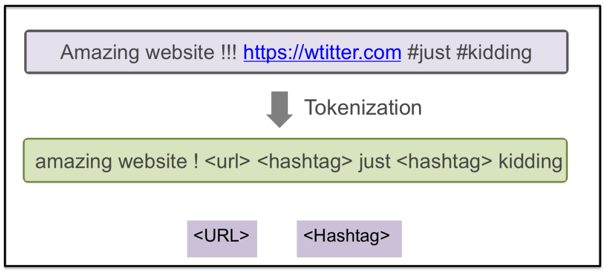
1) Tweet Tokenization: Then we use the ID as the feature for each word to train our word embedding matrix. As a specialized corpus, tweet data contain large amount of micro-blogging features, such as emojis, hand-shorts, hashtags and user mentions [10]. A straight-forward approach to preprocess these data is to consider all above features as noises and remove them from each tweet. However, this may lead to information loss, as this micro-blogging features usually reveal strong sentiment trends. So our approach is to map each kind of above-mentioned feature into a specific keyword using regular expression:

- Emotions: We map all different kinds of emotions in a tweet to the same keyword $< emoji >$.
- Shorthands: We collect most popular shorthands in twitter. For each tweet, we convert the shorthands in it into a phrase revealing its sentimental tendency. For example, *3>* means *love*.
- Hashtags: We seperate the hashtag from its following word, and map the hashtag to a keyword *hashtag*. For example, $\#amazing$ will be $< hashtag > amazing$.
- Mentions: we map all user mentions to a keyword $< user >$.
- Duplications: For each word in a tweet, if there are duplicate letters for more than three times, such as *coooooooool*, we will only keep two letters of them, so the above-mentioned word will change to *cool*.
- URLs: Since URLs are usually irrelevant with the sentiment tendency of the tweet, so we just replace all URLs in tweets with a keyword $< URL >$.

2) Word to Vector Representation: We cannot feed our RNN model with raw tweet text, so a vector representation for tweet is necessary. Especially, we adopt a skip-gram model to get a word-embedding matrix for all words in the Twitter corpus [11], so that the semantic relevance among different words can be captured and used as inputs to the RNN model. We get this matrix by training a single-layer neural network. The input of the model is a single word w_I, and the output is the words in its context $\{w_{O,1}, w_{O,2}, \dots, w_{O,C}\}$ defined by a word window of size C. As shown in Figure 3, x represents the one-hot encoded vector corresponding to the input


@MrL you are so coooooooooool 👍 :-) <3
Tokenization
<user> you are so cool <emoji> <smile> <heart>
<User Name>
<misspelling >
<Emotion>
<Abbreviation>

(a)


Amazing website !!! https://wtitter.com #just #kidding
Tokenization
amazing website ! <url> <hashtag> just <hashtag> kidding
<URL>
<Hashtag>

(b)

Fig. 2: **Twitter Tokenization with respect to different micro-blogging features**

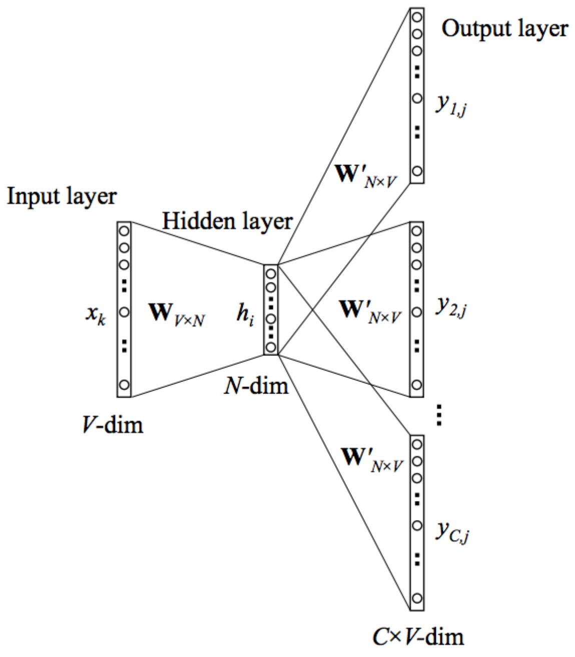


Fig. 3: A skip gram neural network

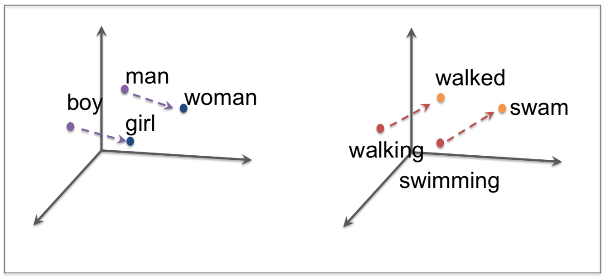


Fig. 4: Vector Representation of words

word in the training text, and $\{y_1, y_2, \ldots, y_C\}$ are the one-hot encoded vectors corresponding to the output words in the training text. The $V \times N$ matrix W is the weight matrix between the input layer and hidden layer whose i^{th} row represents the weights corresponding to the i^{th} word in the vocabulary. This weight matrix is what we call the word-embedding matrix because it contains the vector encodings of all of the words in our vocabulary. We can use this word-embedding matrix to capture the semantic relevance among words, so that two words with close semantic meanings will have smaller distance in the vector space.

B. Recurrent Neural Network

Recurrent Neural Networks (RNN)s are a specialized neural network that consists of multiple copies of the same network cell, as shown in Figure **??**. Due to this recursive structure, it is suitable for processing sequential data [**?**]. For each network cell A_t in the RNN model, its has an input s_t, a hidden state s_t, and generates an output h_t. The output and hidden state are then scaled and feed as inputs to its successor network cell. However, a simple RNNs cannot learn long time dependency, because during the optimization step, the parameter matrix this term tends to vanish or explode very fast due stochastic gradient [**?**]. To solve this challenge, specialized RNNs with gates is proposed and becomes one of the most effective practical models that used for sequential data.

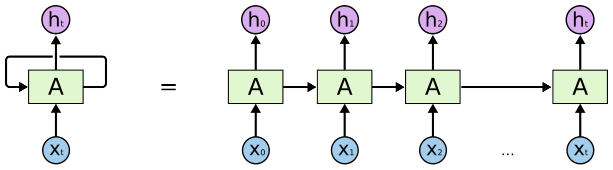


Fig. 5: A Recurrent Neural Network

1) Long Short Term Memory: In our project, we will use a specialized RNN network called Long Short Term Memory (LSTM). Long Short Term memory (LSTM) network is a branch of gated RNNs that can capture long term dependencies when processing sequences of input texts.. It is widely used in various machine learning and NLP applications, such as speech recognition, machine translation, and handwriting generation [**?**] . Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. A LSTM network contains recursive LSTM network cells, each of which consists of the following gates:

- Forget gate f_i:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

  It generates a scale parameter between 0 and 1 for each component in the previous LSTM cell C_{t-1}, based on the current input x_t and previous output h_{t-1}.
- External input gate i_t: the external input gate unit is computed with the following equation:

$$i_t = \sigma(b_i + W_i[h_{t-1}, x_t])$$

- Update gate $\tilde{C}_t$: This gate updates the internal state of the LSTM cell by the following equation:

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- Output gate $h^{(t)}$ and $q_i^{(t)}$: The output $h^{(t)}$ and the output gate $q_i^{(t)}$ are updated using sigmoid function also:

$$\begin{aligned} o_{(t)} &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\ h(t) &= \tanh(C_t) o_{(t)} \end{aligned}$$

Using the above four gates in a LSTM cell, the gradient can flow for long duration , and therefore LSTM can learn long-term dependencies more effectively than normal RNNs.

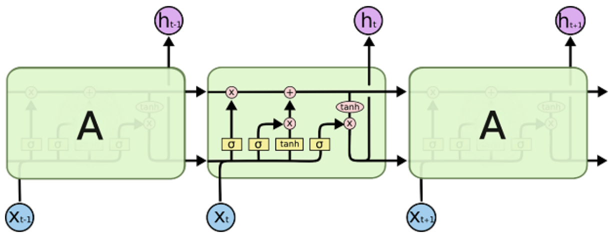
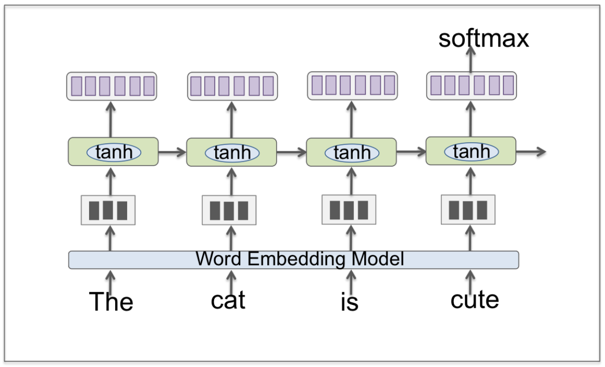
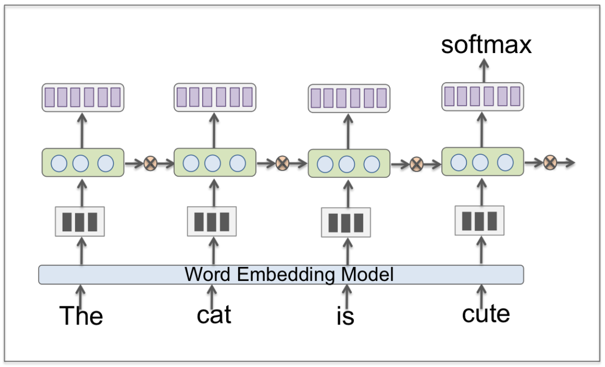


Fig. 6: A Long Short Term Memory network



(a) RNN



(b) LSTM

Fig. 7: **RNN model and LSTM model for sentiment analysis**

IV. Evaluation

We trained and evaluated our sentiment classifier using around 20000 human-labeled tweets, and compare our performance with three baseline sentiment classification approaches. Evaluation shows that our RNN model achieves the highest accuracy compared with baseline models, and has the faster convergence rate compared with models using Convolution Neural Network.

A. Available Dataset

We adopted the Thinknook Dataset [3] for our evaluation. It is composed of tweets from two sources:

- University of Michigan Sentiment Analysis competition on Kaggle [1].
- Twitter Sentiment Corpus by Niek Sanders [4].

This dataset contains $1,578,627$ classified tweets since 2012, each row is marked as 1 for positive sentiment and 0 for negative sentiment. For each sample in the dataset, it contains the tweet ID, the tweet content, and a sentiment label of either *pos* and *neg*. We extract the tweet content and sentiment labels, and separate the huge dataset into multiple batches of the same size. After preprocessing our data, each batch is a three-dimensional matrix representation which can be used to train and test our RNN model.

B. Baseline Models

We adopt three baseline approaches to compare with our model: the first two models are based on bag-of-words approaches, that is, these models do not consider the order of words shown in the document, but consider the frequency for a word to appear in the corpus.

- SVM: For SVM model, we adopt a linear model, and use the Term Frequency - Inverse Document Frequency (TF-IDF) as our input features. TF-IDF is widely used in NLP text classification tasks. It consists of two terms: a term frequency tf and an inverse document frequency idf:
- Naive Bayes : For naive Bayes model, we use the uni-gram as the input features. Before training model, we adopt the same pre-processing approach as our RNN model.
- Convolution Neural Network (CNN): We adopt the CNN model for sentiment analysis proposed by Kim *et al.* [9]. It applies principals from image processing to a tweet's 2 dimensional sentence vector. In this model, we use a single layer CNN.

C. System Implementation

We used deep learning Python modules called *TensorFlow* [2] to implement our RNN model. The core parameters are given in Table 7(a). The core of the model consists of an LSTM cell that processes one word at a time and computes probabilities of the possible values for the next word in the sentence. The memory state of the network is initialized with a vector of zeros and gets updated after reading each word.

TABLE I: RNN model parameters

Parameter	**Value**
batch_size	64
embedding size	64
hidden layer	50
l2 rate	0.1
dropout rate	0.5
learning rate	0.0001
max epoch	30

- **Truncated Backpropagation.** In order to make the learning process tractable, it is important to create an unrolled version of the network which contains a fixed number of LSTM inputs and outputs, known as the number of steps. The model is then trained on this finite approximation of the RNN. In our model, we tune the steps to be 30 instead of 140 which is the maximum tweet limit, as most tweets are short and far less than the maximum limit. Then we can perform a backward pass after each such input block.
- **Optimizing Loss Function.** We aim to minimize the average negative log probability of the target words:

$$\text{loss} = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{target_i}$$

We implement by using *TensorFlow* 's *seq2seq* library. It will generate the weighted cross-entropy loss for a sequence of logits, based on three inputs: the logits, the targets, and the weights.

- **Stacking Multiple LSTMs.** We use multiple layers of LSTMs to process the data so that the output of the current layer will become the input of its succesor. In our model, we use the libray called *MultiRNNCell* to implement three LSTM networks to strike a balance between the training time and the quality of the model output.

D. System Comparison

We use the average accuracy as our metric to evaluate the performance of different models. We explore with different sizes of training data from 5000 tweets to 100000 tweets, and use models trained on different dataset to test coming tweets. In our experiment, we select a set of 20000 tweets as the testing set. For each set of training data, we use the same pre-processing steps to tokenize tweets. For both RNN and CNN, we use the same word to vector transformations.

1) Comparison with traditional models: Our classification model outperforms SVM and Naive Bayes model by over 20% accuracy. We use the same set of testing data, and explored with training data of size ranging from 5000 to 100000, and the performance for traditional models cannot reach 80% even when using the largest size of training data. Figure 8 is the results using different size of training data, and Figure 9.

2) Comparison with CNN model: RNN model outperform CNN in terms of accuracy and convergence rate. We can see from Figure 8 that RNN accuracy is larger than CNN. Moreover, Figure 10 shows the accuracy results of training data when in different epoch times. For each epoch (iteration), we feed the RNN and CNN model with a batch of 64 tweets which are mapped to vector representations. We can see that in ,the training accuracy of RNN becomes stable after 12 iterations. But for CNN, it takes more iterations to achieve robustness (after 26 iteration).

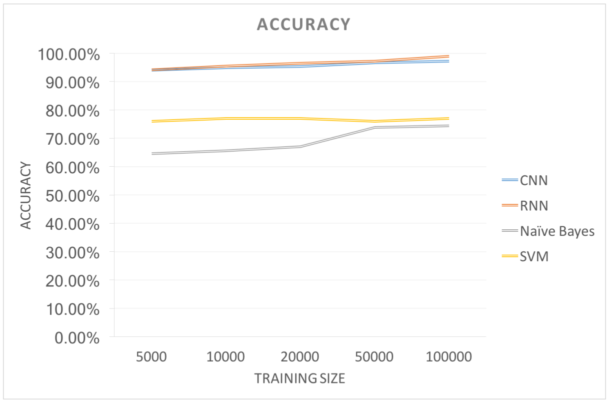


Fig. 8: Model performance with different sizes of training data.

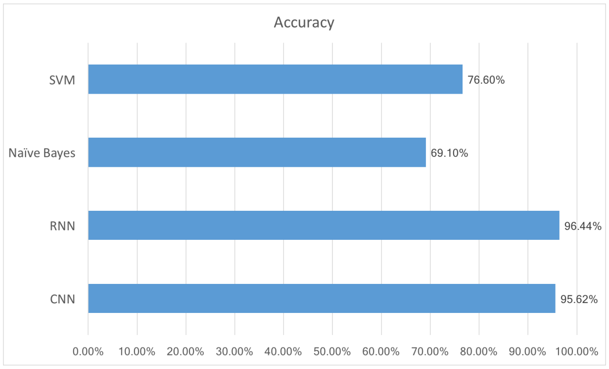


Fig. 9: Average performance

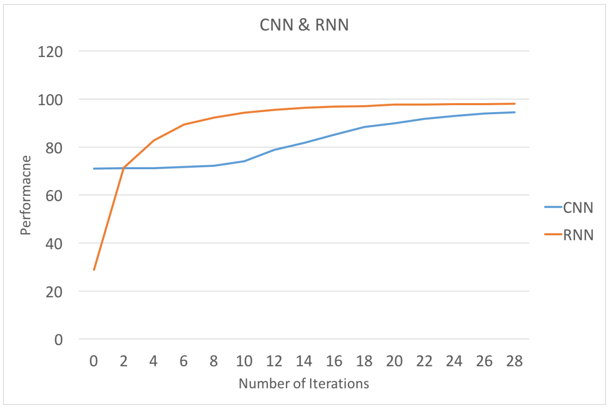


Fig. 10: Convergence rate of RNN and CNN.

V. Conclusion and Future Work

In this project, we implement an RNN model to achieve tweet sentiment classification using word-to-vector features. We also implement three baseline models to compare with our approach. We use a SVM model and Naive Bayes model to compare our results with traditional bag-of-words classifiers, and use a standard Convolution Neural Network model to compare with deep learning classifiers. Our approach outperforms all these baseline models in terms of accuracy and convergence rate.

In the future, we plan to explore with more complex neural network models. Especially, RNNs has various structures and we may benefit from advanced RNN models. For example, Matrix Vector RNN [15] and Recursive Neural Tensor Network [14] already prove it can efficiently classify among movie reviews. These models may further improve the results of tweets sentiment classification. We also plan to explore how twitter features, such as emotions, hashtags and short hands, can be used to improve the performance of prediction on the sentiment label of tweets.

References

[1] Kaggle Dataset. https://inclass.kaggle.com/c/si650winter11. Accessed: 2017-03-21.

[2] Kaggle Dataset. https://www.tensorflow.org. Accessed: 2017-03-21.

[3] Twitter Sentiment Corpus. http://thinknook.com/wp-content/uploads/2012/09/Sentiment-Analysis-Dataset.zip. Accessed: 2017-03-21.

[4] Twitter Sentiment Corpus. http://www.sananalytics.com/lab/twitter-sentiment/. Accessed: 2017-03-21.

[5] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics, 2011.

[6] J. Bollen, H. Mao, and A. Pepe. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *ICWSM*, 11:450–453, 2011.

[7] J. Bollen, H. Mao, and X. Zeng. Twitter mood predicts the stock market. *Journal of computational science*, 2(1):1–8, 2011.

[8] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12), 2009.

[9] Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[10] E. Kouloumpis, T. Wilson, and J. D. Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11(538-541):164, 2011.

[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[12] F. Mosteller and D. Wallace. Inference and disputed authorship: The federalist. 1964.

[13] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, 2010.

[14] R. Socher, B. Huval, C. D. Manning, and A. Y. Ng. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics, 2012.

[15] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer, 2013.

[16] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10(1):178–185, 2010.

[17] H. Wang, D. Can, A. Kazemzadeh, F. Bar, and S. Narayanan. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics, 2012.