LSTM for Sentiment Analysis on Twitter

Trapit Bansal **Kate Silverstein** **Jun Wang**

Abstract

abstract goes here

1 Introduction

In the last few years, microblogging has become a very popular communication tool in people's social life. On microblogging platforms, such as Twitter[1] and Facebook[2], a diverse range of people are attracted to post short sentences, images, and video links to share life issues and opinions. This popularity results in enormous amount of information covering a wide range of topics on the brands, products, politics and social events. Such data is a valuable and efficient source for marketing and social studies. Sentiment analysis on microblogging has obtained special interest, because it determines the attitude of a user with respect to some topic or product and thus provides provide convincing information For example, it may help manufacturing companies to know how people like their product (or service), what would people prefer.

In this paper, we focus on using sentiment analysis on Twitter. Twitter is an extremely popular microblogging platform which allows people to post messages of up to 140 characters. Because of the short nature of tweets, people often post twitter messages (called Tweets) frequently while attending events like product launches, movie premiers, music concerts or just to express their opinion on a trending or current topic. As such, they can be a valuable source of public opinion or feedback [15, 1, 2]. Realizing this importance, analyzing sentiment of Tweets has been a recurring task in the SemEval [3] competitions.

Although there exists plenty of work on text classification, some unique characteristics of tweets present special challenges for sentiment analysis: 1) Tweets are short in length. There is a limitation of 140 words for each tweet which makes analyzing them challenging [12]; 2) The language used in tweets is very informal with misspellings (often intentional, like different spellings of ”cool”: coool, coolll, cooooolll!!), new words, slangs, and URLs; 3) The number of tweets increases at a very fast pace, and with new data comes new words, new trends in using abbreviations, which lead to a frequent problem of out-of-vocabulary words 4) Special symbols and their combinations are often used, like emoticons and hashtags.

Keeping in view the challenges posed by the nature of tweets, in this paper, we propose to use bi-directional Long Short-Term Memory (LSTM) networks which operate at *character-level input* and make predictions at the tweet-level. Such networks naturally handle the problem of very large vocabulary sizes and the presence of sub-word information, without having to keep many trained embedding which would result from keeping very large vocabularies. We find that the model interestingly performs **better** than equivalent LSTM model which operates at the word level (even when initialized with a variety of pre-trained embeddings). We also compare our results with Dynamic Convolutional Neural Network (DCNN) [7] which has shown state of the art performance on Twitter sentiment classification [17]. We explore the properties of the character LSTM model and find that it learns to find meaning in composition of characters and can effectively relate sub-word information (like presence of multiple exclamation marks) to the sentiment of the tweet.

[1] `https://twitter.com/`

[2] `https://www.facebook.com/`

[3] `http://alt.qcri.org/semeval2016/task4/`

In the remaining of this paper, we first introduce the related work, the dataset, and the evaluation methodology. We then describe the model we proposed. Finally, we discuss the experiment results and point out possible directions for future research. (Please revise this part.)

2 Related Work

Twitter sentiment analysis is increasingly drawing attention of researchers in recent years. Given the length limitations on tweets, sentiment analysis of tweets is often considered similar to sentence-level sentiment analysis [10]. However, phrase and sentence level approaches can hardly define the sentiment of some specific topics. Considering opinions adhering on different topics, Wang et. al. [19] proposed a hashtag-level sentiment classification method to generate the overall sentiment polarity for a given hashtag. Recently, following the work of [13] some researchers used neural network to implement sentiment classification. For example, Kim [9] adopted convolutional neural networks to learn sentiment-bearing sentence vectors, Mikolov et al. [14] proposed Paragraph vector which outperformed bag-of-words model for sentiment analysis, and Tang et. al. [18] used ConvNets to learn sentiment specific word embedding (SSWE), which encodes sentiment information in the continuous representation of words. Furthermore, Kalchbrenner [7] proposed a Dynamic Convolutional Neural Network (DCNN) which uses dynamic k-max pooling, a global pooling operation over linear sequences. Instead of directly applying ConvNets to embeddings of words, [20] applies the network only on characters. They showed that the deep ConvNets does not require knowledge of words and thus can work for different languages. LSTM [6] is another state-of-the-art semantic composition models for sentiment classification [11]. Similar to DCNN, it also learns fixed-length vectors for sentences of varying length, captures words order in a sentence and does not depend on external dependency or constituency parse results.

2.1 Dynamic Convolutional Neural Networks (DCNN)

We briefly review the architecture of DCNN [7] which has shown state of the art performance for sentiment classification on Twitter. The winning entry for SemEval15 [17] task on Twitter sentiment classification also used DCNN. Figure 1 summarizes the architecture.

When used for sentiment classification on Twitter, the input to the DCNN is a matrix of word embeddings for each word in the tweet. For example, if the tweet consists of s words then the input to the DCCNN is:

$$S = \begin{bmatrix} | & | & \dots & | \\ w_1 & w_2 & \dots & w_s \\ | & | & \dots & | \end{bmatrix}_{k \times s}$$

where each $w_i \in R^k$ is a k-dimensional dense word embedding [14]. The architecture consists of multiple layers of convolutions and max-pooling on top of the input matrix, followed a fully connected layer which is input to a softmax. The convolutions are of type *wide-convolutions* of one-dimension. For example, for the input matrix $S \in R^{k \times s}$, a wide-convolution filter operating on S will consist of convolution weights $m \in R^{k \times c}$ and will result in a matrix having dimension $k \times (s + c - 1)$. Note here c is the convolution filter width, which is a hyperparameter. The max-pooling operations presented in [7] are different from the regular max-pooling. They present *dynamic k-max pooling*. k-max pooling takes the top k maximum activations as opposed to just the maximum activation and the value of k is selected dynamically based on the following formula: $k_l = \max\left(k_{top}, \lceil \frac{L-l}{L} s \rceil\right)$, where l is number of current convolution layer, L is total number of convolutions and k_{top} is a fixed hyperparameter. Note that while [7] used multiple layers of convolutions and max-pooling, subsequent work found that using a single layer of convolution and max-pooling gives similar results [9] [17].

2.2 Recurrent Neural Networks with Long Short Term Memory (LSTM)

Recurrent Neural Networks (RNNs) are a class of artificial neural networks used for modeling sequences. RNNs are highly flexible in their use of context information as they can learn what part of the input sequence to store to memory and what parts to ignore. They also allow modeling of various regimes of sequence modeling as shown in Figure 2. Please refer to [4] for a comprehensive review of sequence modeling using RNN.

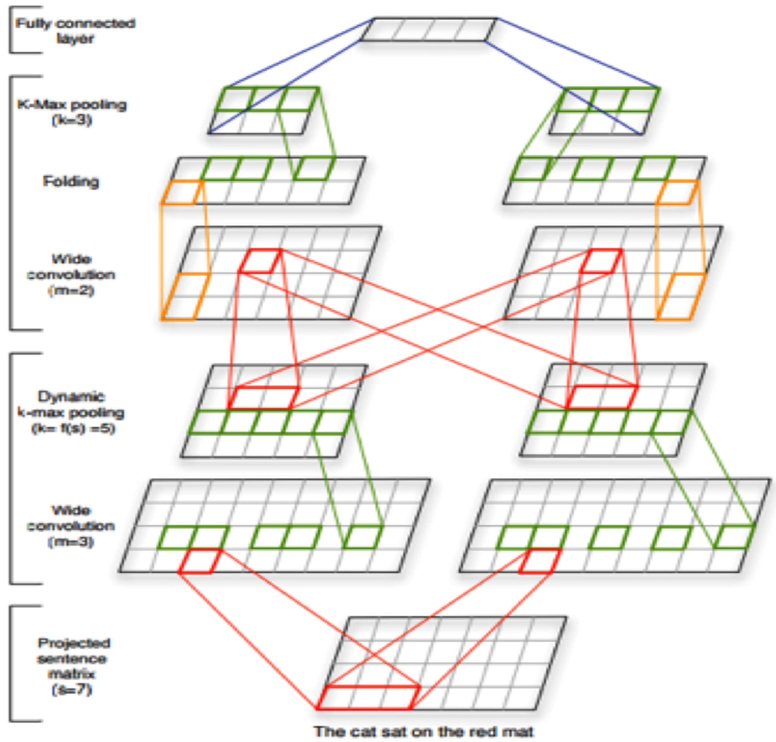

Figure 1: Dynamic Convulational Neural Network of [7] (Source: [7])

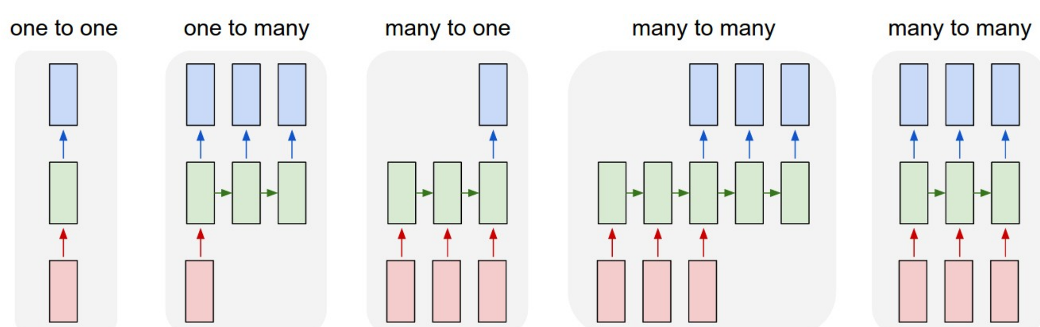

Figure 2: RNNs allow modeling of multiple types of input and output sequences (Source: [8])

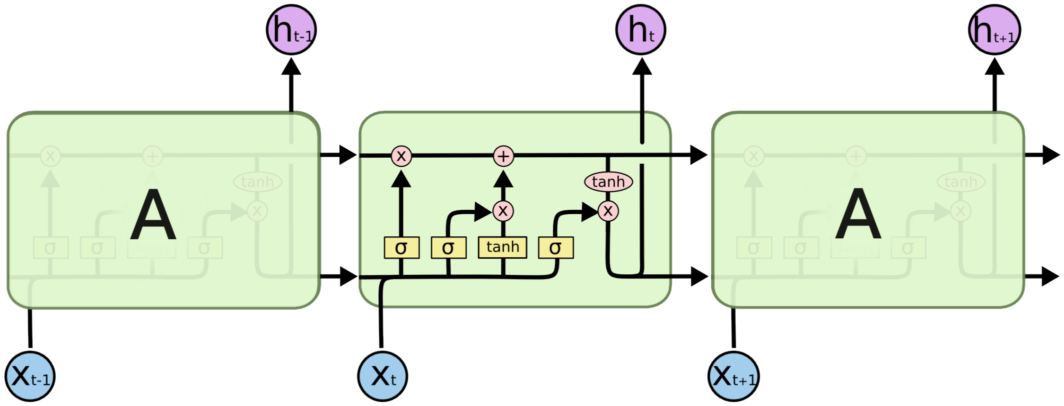

Figure 3: The repeating module of LSTM. x_t is the input as time t and h_t is the output from the LSTM output gate at time t. The top horizontal line corresponds to the cell state and the bottom line corresponds to the hidden state (both of which are recurring states). (Source: [16])

One of the short comings of RNN is that it is very difficult to store information over long sequences because of problems due to vanishing and exploding gradients as explained in [5]. *Long Short-Term Memory (LSTM)* [6] are designed to remedy this and store information over larger input sequences. They achieve this using special "memory cell" units. Figure 3 shows the architecture of this cell which consists of an input gate, a forget gate, an output gate and a recurring cell state. Refer to [16] for a gentle introduction to LSTM and to [4] for a more comprehensive review and applications.

3 Twitter Sentiment Analysis using Character LSTM

In this section we present LSTM models for sentiment analysis of twitter messages. We explored different architectures of LSTM networks which operate at word-level or character-level input. The basic architecture of the network is shown in Figure 4. We explain the model architecture considering the character input. Given a tweet as a sequence of characters $X = \{x_1, \dots, x_{140}\}$, the task is to predict the sentiment of Tweet as being *positive* (1) or *negative* (0). Each x_i is a one-hot encoding of either the character or the word. The LSTM models then take this one-hot encoding and convert them into either a *character embedding* or a *word embedding* depending on whether the input is characters or words. The embeddings can be randomly initialized and learned jointly with other model parameters.

The model consists of a bidirectional LSTM layer over this character input X. The hidden layer activations obtained from the forward and backward LSTM layer are averaged at *each* character which serves as the input to the *second layer* of LSTM. The second layer of LSTM takes as input this sequence of hidden layer outputs from first layer and encodes a representation of the tweet at the last element of the sequence (the hidden layer activation of the last LSTM unit). This output is then fed into a softmax layer which classifies the Tweet as being positive or negative. The model is trained using categorical cross-entropy.

Note that it is possible to have multiple levels of granularity in predictions, like positive, "netural" and negative, or even finer. We restrict ourselves to two classes to be able to compare with the state of the art published results [7].

The same model can be used with either character input data or word input data. Note that when used with word input, it is common to initialize the models with pre-trained word embeddings [14]. We explore multiple such initializations in our experiments. For the character input model, the embeddings are always initialized randomly, since pre-trained character embeddings are not yet available.

We explored other variants of the model like have single-directional LSTM, having a single layer as opposed to multiple layers, using concatenation instead of averaging after first layer. We also tried an ensemble LSTM which takes both characters and words as separate inputs which are combined at

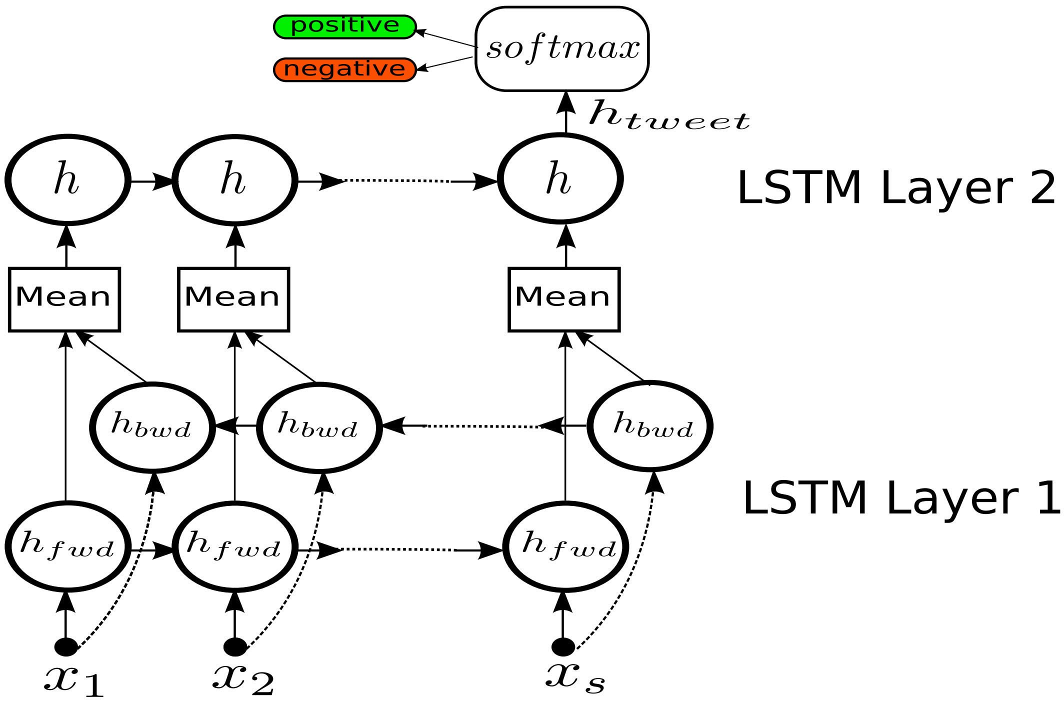

Figure 4: The bi-directional LSTM model used for Twitter sentiment analysis. Each circular node represents an LSTM cell. The input to the model is the sequence x.

the last stage by concatenation before feeding to the softmax. Unfortunately, none of these models gave good preliminary results and in the experiments we focused on just the model of Figure 4 with either character or words as inputs.

4 Experiments

See Table 1 for awesome results

Table 1: Sample table title

PART	DESCRIPTION
Dendrite	Input terminal
Axon	Output terminal
Soma	Cell body (contains cell nucleus)

5 Conclusion

Acknowledgments

References

References

[1] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[2] Johan Bollen, Alberto Pepe, and Huina Mao. Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *arXiv preprint arXiv:0911.1583*, 2009.

[3] Alec Go, Richa Bhayani, and Lei Huang. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12, 2009.

[4] Alex Graves et al. *Supervised sequence labelling with recurrent neural networks*, volume 385. Springer, 2012.

[5] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies*. A field guide to dynamical recurrent neural networks. IEEE Press, 2001.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[8] Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks. `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`.

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

[10] Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. Twitter sentiment analysis: The good the bad and the omg! *Icwsm*, 11:538–541, 2011.

[11] Jiwei Li, Dan Jurafsky, and Eudard Hovy. When are tree structures necessary for deep learning of representations? *arXiv preprint arXiv:1503.00185*, 2015.

[12] Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 889–892. ACM, 2013.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[15] Brendan O'Connor, Ramnath Balasubramanyan, Bryan R Routledge, and Noah A Smith. From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11(122-129):1–2, 2010.

[16] Christopher Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

[17] Aliaksei Severyn and Alessandro Moschitti. Unitn: Training deep convolutional neural network for twitter sentiment classification.

[18] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1555–1565, 2014.

[19] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM, 2011.

[20] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657, 2015.