International Journal Of Advance Research, Ideas And Innovations In Technology

ISSN: 2454-132X
Impact Factor: 6.078
(Volume 9, Issue 5 - V9I5-1448)
Available online at: https://www.ijariit.com

Utilizing LSTM Neural Networks for Sentiment Analysis of Tweets

Manan Gangwani
manangangwani06@gmail.com
Podar International School, Maharashtra, Mumbai

ABSTRACT

Deep Neural Networks are considered as one of the most powerful machine learning methods of recent times. Recurrent neural networks, including LSTM variations, exhibit exceptional performance in sequence-oriented assignments, while also falling within the domain of autoregressive models, wherein forecasts are tied to the historical input context. In this paper, we experiment with LSTM for twitter sentiment analysis. Leveraging advances in Natural Language Processing (NLP), we show the efficacy of our algorithm with extremely competitive results.

Keywords — *Neural Networks, Long-Short Term Memory, Sentiment Analysis*

1. INTRODUCTION

NLP exploits natural language text or speech with computer application, in order to achieve something applicable for our daily lives. Combined with com putational linguistic and speech technology, NLP is perpetually being used as a major component in Human Language Technologies, which aims to develop and implement appropriate tools for computer systems to understand natural languages and to execute desired tasks. Sentiment analysis focuses on understanding how emotions, opinions, and sentiments of people are expressed in texts. Sentiment analysis is not only one part of study for Human Language Technologies, but also one of the most active research subject of NLP. As social networks are heavily used by society, e.g. Twitter, to express opinions and emotions, the needs of leveraging advanced study of sentiment analysis starts to arise, especially for business benefits[1].

In this endeavor, we harness *LSTM Neural Networks* in tandem with *Global Vectors for Word Representation (GloVe)* for sentiment analysis. The efficacy of LSTM networks extends to diverse sequential tasks, including language un derstanding [?] and motion pattern analysis [?]. While GloVe outperforms the Skip-Gram Model [3], we delve into tweet characteristics, followed by an insight into our preprocessing methodologies. We introduce the GloVe embedding that facilitates word-to-vector mapping, while outlining our classifier's design. Then, we lay out the specifics of training parameters, culminating in the presenta tion of experimental outcomes. Ultimately, we summarize findings and propose strategies for accuracy enhancement.

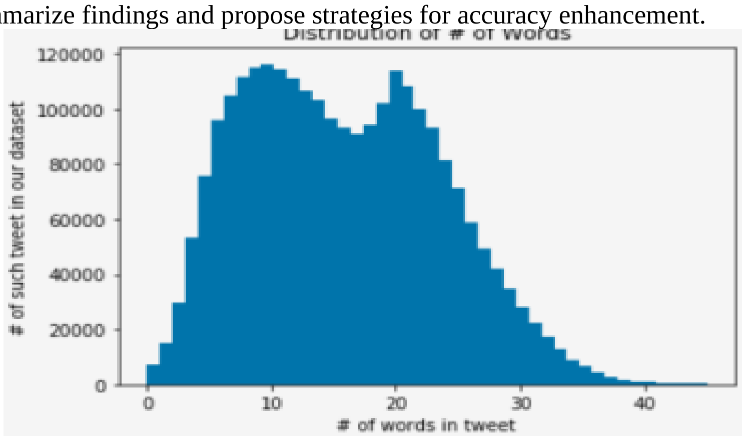


Fig. 1: Distribution of # of words

2. OBSERVATION ON DATASET

We use the Twitter large dataset of 2,500,000 tweets which consists of 1,250,000 positive and negative tweets each. Each tweet has length at most of 140 char acters, and usually has positive or negative smiley. Furthermore, most of tweets consist of less than 45 words (Fig. 1). In addition, we observe that language of tweets varies; mainly English, with some other languages like French, Malay, and so on. We clarify the accuracy of our model for sentiment analysis with twitter test dataset which contains 10,000 unlabelled tweets.

3. DATASET PREPROCESSING

Before proceeding into the training step, we perform preprocessing by remov ing some meaningless words and expanding or replacing words with more useful terms. The reason why we transform these words to specific words will be ex plained in section IV.

3.1 Words and Characters Removal

Based on exploratory analysis of the training dataset, we observe some mean ingless words and characteristics that give almost no information to sentiment analysis. Thus, we remove:

Utilizing LSTM Neural Networks for Sentiment Analysis of Tweets 3

– *Numbers*

Any form of numbers as a word, such as timestamps of 23:55 and 1:20, is quite pointless for classifying the sentiments.

– *Repeating Characters*

In tweets, we often see something like *'excitedddddd'* or *'thaaanks'*. We remove the redundant characters so that they will be considered as *'excited'* and *'thanks'* while doing GloVe embedding.

3.2 Words Transformation

In addition to removing some words and characters, we also extract some com ponents in tweets to give more meaningful representations. Therefore, we:

– *Expand English Contraction*

In written English, contraction is frequently used, such as "I'm", "You're", "He isn't", and so on. In order to feed the words into GloVe embedding, we expand these words into "I am", "You are", "He is not", and so on. – Reduce Repeated English Punctuation We reduce repeated punctuations in tweets, e.g. '!!!'.

– *Highlight Sentiment Words*

Some words in English tend to have explicit emotional feelings. For instance, "*convenient*" is more likely positive while "*abuse*" has negative tendency. To achieve this, we utilize opinion lexicon datasets [8], which contain dataset of positive and negative words in English, and add "*positive*" or "*negative*" word before the actual word.

– *Split Hastags(#) into Words*

Hashtag is commonly used in tweets and it is regularly used to emphasize tweets meaning. However, splitting a hashtag into undetermined number of words is a difficult task. For example, "*#meaningless*" can either be "*mean ingless*" or "*meaning less*". To overcome this issue, we make use of word dictionary from small subset of Wikipedia and predict the words in hashtag according to frequency (pick most frequent word as possible). By dynamic programming, we split the hashtag before we map words into vectors using GloVe embedding.

– *Transform Emojis into Special Words*

Based on the mouth of emoji, we can sometimes know the sentiment of this emoji. For example, ':)' usually refers to positive expression and ':(' relates to negative statement. Hence, we transform some emojis into some special words, e.g. *'<lolface>'* and *'<sadface>'*. However, the sentiment of emoji like ':-o' is not obvious, so we use *'<neutralface>'* in this case. As we use the GloVe embedding dataset from Stanford NLP group, we need to transform some of the emojis into these special words in tweets because not all emojis are available in this dataset.

4 GLOVE

The GloVe [9] algorithm maps a word to a vector. The theory of GloVe is based on co-occurrence statistics. For instance, the co-occurrence probability of 'ice' and 'solid' is higher than that of 'ice' and 'gas'. Machines are employed to analyze multiple articles, quantifying the frequency of co-occurrence between pairs of words. The optimization process involves subgradient descent to minimize the ensuing function.

$_{J=}X^{V}\ i,j=1$

$f(X_{ij})(w^{T}_{j}\tilde{w}_{j} + b_{i} + \tilde{b}_{j} - logX_{ij})^{2}$

where f is the weighting function, V is vocabulary, X_{ij} is co-occurrence proba bility, and w and b are parameters to be trained. We transform and emphasize words to sentiment words in our preprocessing so that we can more effectively use the pre-trained word vectors for tweets from Stanford NLP group [11]. This GloVe dataset includes all the transformed words, such as 'positive', <sadface>', and represents each frequent word with a 200-dimension vector. With the pre trained word vectors, we generate a 40x200 matrix for each tweet as an input to

LSTM:

word vector of 1*st word*

word vector of 2*nd word*

...

word vector of 39*th word*

word vector of 40*th word*

If the number of words is less than 40, then we pad 0's to the matrix. There are 2 reasons why we choose 40 as the number of rows. First is that the number of words in 99.93% of preprocessed tweets is less than or equal to 40 (Fig. 2), and the second reason is that it leads to higher accuracy (Table 3)

5. ARCHITECTURE

The general architecture used in this this experiment is shown in figure 3. At each timestep, the LSTM [7] receives a 200-dimensional GloVe embedded word and updates its parameters via the following recurrence equation:

input gate $i_t = sigm(W_i \cdot [x_t, h_{t-1}] + b_i)$

forget gate $f_t = sigm(W_f \cdot [x_t, h_{t-1}] + b_f)$
output gate $o_t = sigm(W_o \cdot [x_t, h_{t-1}] + b_o)$
input modulation gate $g_t = tanh(W_g \cdot [x_t, h_{t-1}] + b_g)$
memory unit $c_t = f_t \circ c_{t-1} + i_t \circ g_t$
hidden unit $h_t = o_t \circ tanh(c_t)$

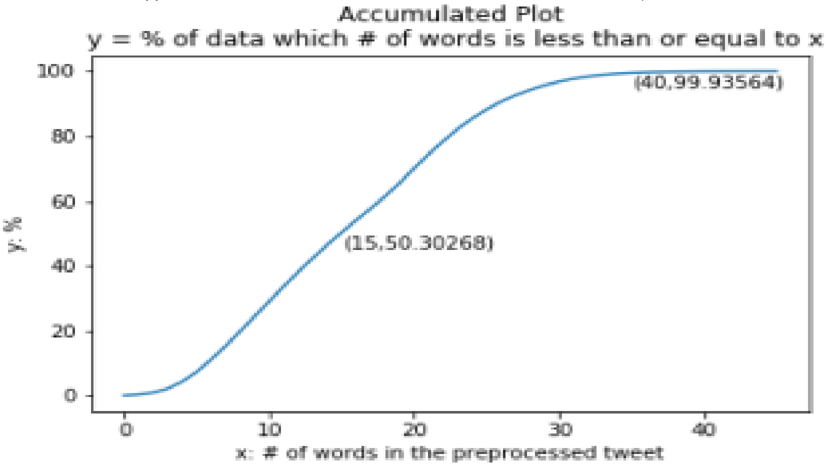


Fig. 2: Accumulated Plot of # of Words

Intuitively, the use of sigmoids at the gates allow LSTMs to control the flow of information through the unit by looking at the current input and past time steps.Our selection of LSTMs is rooted in their capacity to capture extended dependencies, a crucial aspect where conventional RNNs falter due to the chal lenge of vanishing or exploding gradients [4, 6]. In our implementation, we pass a zero vector to the LSTM over the remaining timesteps for tweets that contain less than 40 words. The output at t=40 is then forwarded through 4 fully con nected layers with [512,512,512,2] units before applying a sigmoid operation at the end for classification.

6. EXPERIMENTS

6.1 Training

We initialize all parameters using the glorot uniform initializer [13]. Optimization is performed using stochastic gradient descent with a batch size of 1000. We use the learning scheme of RMSProp [12] with a base learning rate of 5e-4, a decay of 0.9 and with no momentum. We unroll the LSTM with 1024 units for 40 time-steps and train the model for 10 epochs. The dataset is split with ratio of 90% for training and 10% for validation. To avoid over-fitting, we use dropout [10] on all fully connected layers with a keep probability of 0.5. We transform the binary labels to a one-hot encoding. The experiments are run in a machine

Embedding

0.1 0.2 0.5 0.9

Input
Preprocessing

Non-Linear Classifier

we're lovingggg
machine learning!!

we

Sequence Learning

we are <positive>
loving machine
learning!
are

GloVe

Fig. 3: Twitter Sentiment Analysis Pipeline

with 2 NVIDIA Titan X and takes approximately 4 hours of training time. We use the Tensorflow library [2] to develop the model.

6.2 Results

In the first experiment, we evaluate the architecture described above against several baselines: Naive Bayes and a Decision Tree using the scikit library [5] with default parameters. Since all these algorithms require each sentence to be a feature vector, we flattened the glove embedded matrix of each word before concatenating the words to form a vector of length 8000. As expected, the per formance of the LSTM surpasses all these models by a significant margin as in Table 1. We speculate that this is due to the fact that these algorithms assume no dependence among the inputs; their output is simply a non-linear combi nation of all input variables. LSTMs on the other hand take advantage of the sequential structure present in language. These findings thus further corroborate the necessity of sequential processing in NLP.

In the second experiment, we study the effect of different configurations of our architecture to the accuracy. We first vary the number of fully connected layers while keeping the LSTM timesteps fixed at 40. To avoid having to experiment with an excessively large number of combinations, we simply fix each layer to have 512 units with the final classification layer at 2 units. The results have been summarized in table 2. A notable observation emerges when transitioning from 3 to 4 layers, showcasing the most substantial enhancement. This becomes intuitive when we perceive the LSTM output as a feature vector embodying the sentiment

of non-linearly separable Twitter posts. The potency of fully connected strata escalates with heightened depth. Conversely, a decline in accuracy is evident upon surpassing 4 layers.
Next, we fix the fully connected layers at 4 layers with [512,512,512,2] units and vary the LSTM timesteps. Refer to table 3. It can be observed that reducing the number of timesteps below 40 results in a decrease in performance. This is due to the fact that the sentiment of some tweets can only be deduced towards Utilizing LSTM Neural Networks for Sentiment Analysis of Tweets 7

Method	Accuracy %
Naive Bayes	62.10
Decision Trees	67.70
LSTM	88.03

Table 1: Validation accuracies of the different methods

FC Layers	Accuracy %
1	87.99
2	88.11
3	88.32
4	91.02
5	89.03

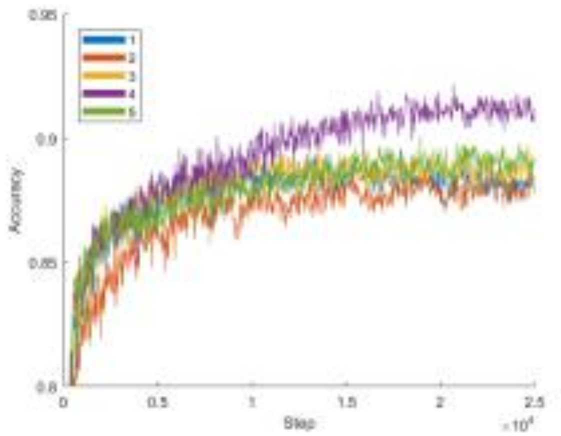
Table 2: Validation accuracies with varied number of fully connected layers. Each layer except the final classification layer contains 512 units. The architecture with only 1 layer would thus have only 2 units.

The end of the sentence. It is also fascinating to note more timesteps with 45 results in a huge decrease in performance. We observe the accuracy oscillating about 65% over the 10 epochs. It is uncertain if the number of timesteps resulted in the architecture is stuck in a local minima in early training. Lastly, we display the validation accuracy plots (figure 4) for our second experiment and we observe that there is no clear sign of overfitting.

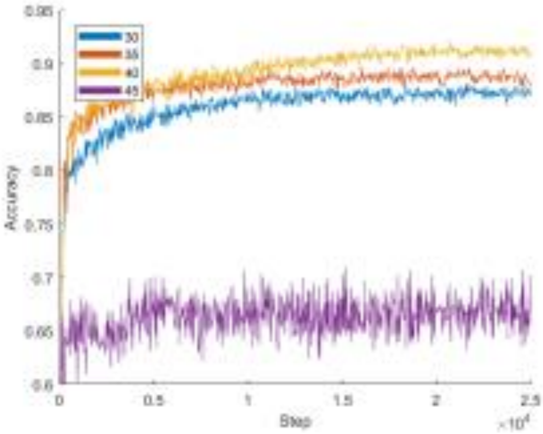
7. CONCLUSION

We have presented a model for twitter sentiment analysis using LSTM neural network. In this project, we figure out that the challenge in preprocessing tweets is mostly related to variety form of word or speech, e.g. 'idk' is 'i do not know'. We do not investigate the following in our experiments but we believe it may improve the performance. In addition, we also do not analyze the structure of sentences. For example, although we map 'hate' to 'negative', the sentence could have been 'I do not hate', so it actually has positive sentiment. Then, we note that our implementation requires the LSTMs to receive a fixed number of words. Improvements in accuracy might be possible if we allow the LSTM timesteps to vary according to the number of words in each tweet.

8 M. Gangwani



(a) Validation accuracy with varied number of fully con nected layers (shown in legend)



(b) Validation accuracy with varied LSTM timesteps (shown in legend)

Fig. 4: Validation accuracy plots for (a) varied numbers of fully connected layers and (b) varied LSTM timesteps

Utilizing LSTM Neural Networks for Sentiment Analysis of Tweets 9

Timesteps	Accuracy %
45	67.51
40	91.02
35	88.16
30	87.35

Table 3: Validation accuracies with varied LSTM timesteps. All models have 4 fully connected layers.

5. REFERENCES

[1] A Pak, P.P.: Twitter as a corpus for sentiment analysis and opinion mining. LREC (2010)
[2] Abadi, M., et al.: Tensorflow: Large-scale machine learning on heterogeneous dis tributed systems. LREC (2010)
[3] et al, T.M.: Distributed representations of words and phrases and their composi tionality. NIPS (2013)
[4] Bengio, Y., Patrice Simard and, P.F.: Learning long-term dependencies with gra dient descent is difficult. IEEE Transactions on Neural Networks (1994) 5. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. ECML PKDD Workshop: Languages for Data Mining and Machine Learning (2013)
[5] Hochreiter, S., Bengio, Y., Paolo Frasconi and, J.S.: Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. A field guide to dynamical recurrent neural networks (2001)
[6] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation (1997)
[7] Hu, M., Liu, B.: Mining and summarizing customer reviews. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Min ing (KDD-2004) (2004)
[8] Jeffrey Pennington, R.S., Manning, C.D.: Glove: Global vectors for word rep resentation. Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
[9] Srivastava, N., Hinton, G., Krizhevsky, A., Ilya Sutskever and, R.S.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research (2014)
[10] Stanford NLP group, Jeffrey Pennington, R.S., Manning, C.D.: glove.twitter.27b.200d.txt. https://nlp.stanford.edu/projects/glove/ 12. Tieleman, T., Hinton, G.E.: Lecture 6.5-rmsprop: Divide the gradient by a run ning average of its recent magnitude. COURSERA: Neural Networks for Machine Learning (2012)
[11] Xavier Glorot and, Y.B.: Understanding the difficulty of training deep feedforward neural networks. Proceedings of Machine Learning Research (2010)

