LSTM Sentiment Classification with Sentiment140

April 2017 - UW Madison

Cormick Hnilicka
UW Madison, Sr
cvhnilicka@wisc.edu

Garritt Moede
UW Madison, Sr
gmoede@wisc.edu

Yazeed Sabri
UW Madison, Sr
ysabri@wisc.edu

ABSTRACT

We aimed to explore the domain of sentiment analysis with multiple Long Short Term Memory (LSTM) Recurrent Neural Network architectures. LSTM cells are well suited to classify over time series related data for they provide a constant stream of relevant information through recurrence, while at the same time forgetting irrelevant information. We make use of LSTMs to predict the sentiment of an arbitrary tweet as either positive or negative. Sentiments classification is could be useful for companies in order to assess people's sentiment about their brand, products, and services. We use the Sentiment140 dataset, it consists of tweets that were collected and labeled using emoticons that appeared in tweet. These emoticons severe as noisy labels. We also preprocessed the tweets in order to limit word embedding entries in our network. Using various LSTM architectures, our models were able to reach accuracies of over 80%.

INTRODUCTION

Since its creation in 2006, Twitter[1] has rapidly risen to a social networking powerhouse that circulates millions of microblogs each day for its ever growing population of over 300 million active users [6]. These microblogs, often referred to as "tweets," consist of opinions, reactions, and reviews to thousands of products, brands, social gatherings, and politics. All these activities make up an extremely large data source for analyses. Results of such analyses could be used in a number of fields. As an example they could be applied to various markets for more efficient marketing.

What differentiates tweets from traditional reviews is their length and shelf life. According to research conducted by Moz's Peter Bray [3], the average tweet's lifespan lasts about 18 minutes. This in combination with Twitter's 140 characters limit per tweet, lead people to often microblog about events they are attending such as new movies, product launches, concerts, as well as other social events.

This magnitude of content makes manual sentiment analysis of tweets tedious and close to impossible. As a result, using deep learning to automate sentiment analysis has gained popularity as means of automating such a process. Mass sentiment analysis is also highly demanded by governments.

Given the short amount of time we had, doing our own data collecting was going to prove difficult, as manually labeling tweets with sentiments would take countless hours. Luckily, [2] previously collected 1,600,000 tweets to analyze their sentiments using various machine learning algorithms such as Naive Bayes, Maximum Entropy, and SVMs. In order to construct their, and ultimately our, dataset, they used a distant supervision approach in gathering their tweets. Twitter's [1] API takes keywords as queries and returns up to 100 tweets containing that query. [2] used the API to collect tweets containing ":)" and ":(" emojis. As explained in [2], tweets that contained they ":)" emoticon were said to have a positive sentiment associated with the tweet while tweets containing the ":(" emoticon were associated with a negative sentiment.

For our approach, we use a sample of the Sentiment140 dataset to construct a word2vec model to serve as our vocabulary. We then use that vector vocabulary to reconstruct tweets by concatenate each word vector in the tweet. From there on we use Keras to build our various models to train and test.

PROBLEM DEFINITION

1. **Preprocessing**

To adequately train our models so that a word in a tweet was uniquely represented, we utilised pre-trained word embeddings using the Word2Vec approach [8].

Our embedding vocabulary was created from the entirety of our training and testing data set to ensure the capture of most words that appear in tweets. However, due to the unique properties of the twitter language model, many strings that should represent the same word actually mapped to different embedding vectors. Take for example, the words '*hunnngry'* and *'hunnnnnnnnngry'*. Both words represent the word *'hungry'* with some exaggeration miss-spelling, but would be mapped to different embedding vectors.

To combat this issue, we performed some feature reduction preprocessing over our entire dataset before both building our embedding vocabulary and training. These feature reduction techniques are outlined in the following table.

TYPE	**ORIGINAL**	**PROCESSED**
Username:	@ede0m	USERNAME
Link/Url:	https://fbi.gov	URL
Elongated letters:	**'***hu<u>nnnnn</u>gry***'**	**'***hu<u>nn</u>gry***'**

In addition to this feature reduction, we also down-sampled the processed training dataset from 1,600,000 tweets to 500,000 tweets in order to train in a reasonable amount of time.

2. RNN's and LSTM's

Natural language processing is a sequencing modelling problem by natural which is why we turned to Recurrent Neural Networks (RNNs) initially. RNNs are a category of neural networks that excel at analyzing sequential data. Their architecture allows for flexibility in analyzing context information as they can distinguish parts of the input sequence to keep as well as parts they can forget [7].

However, one can improve further when using natural language processing in sentiment analysis by using a Long Short-Term Memory (LSTM) network. Traditional RNNs may perform well on sequential data in general but they begin to have difficulty storing information. This difficulty happens when the sequence is very long due to vanishing as well as exploding gradients [4]. Further explanation into RNNs can be found at the following paper [7]. LSTM networks are designed to combat this issue and efficiently store information over larger input sequences by utilizing special "memory cells". The architecture of a LSTM memory cell consists of an input gate, a forgot gate, an output gate, and finally a recurring cell state. Further

information regarding LSTM networks can be found at [5].

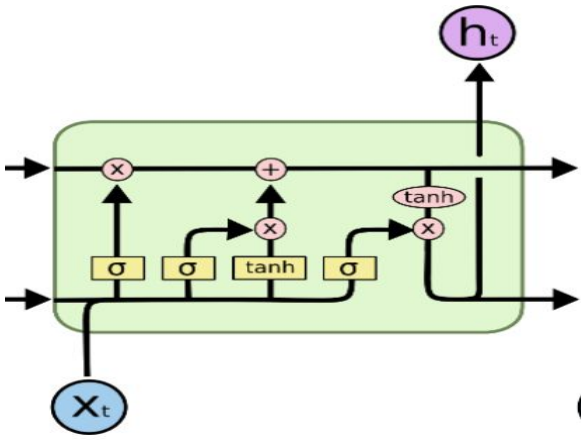


Figure 1: A typical memory cell found in a LSTM. x_t represents the input at time t and h_t is the output from the cell at time t. The line across the top of the cell holds the cell's state and the bottom line represents the hidden state. [5]

3. Our Best Architecture

Our best network consisted of one layer of 100 LSTM memory cells as well as a dense output layer. It is depicted below in **Figure 2.** The network received embedded word vectors as input and output a classification between 0 and 1. Results closer to 0 have a more negative sentiment while results closer to 1 have a more positive sentiment.

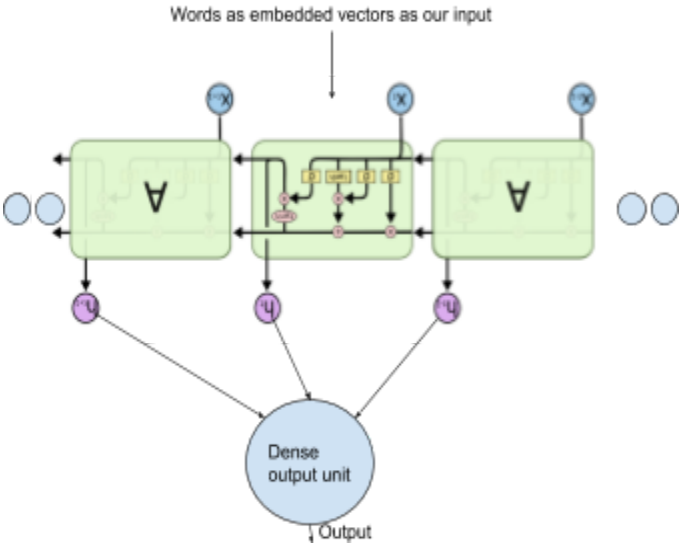


Figure 2: Our best network consisted of one layer of LSTM memory cells, a dense output layer consisting of one unit, and an input layer that consisted of embedded vector representation of the words. LSTM image found from [5].

EXPERIMENTAL EVALUATION

METHODOLOGY

Our overall goal in this experiment was to test various LSTM network architecture accuracies against a fully connected, traditional deep neural network.

Our null hypothesis is then stated as the following. *LSTM network architectures do not have any substantial benefits for predicting the sentiment of tweets compared to traditional deep neural networks.*

To test this, we built a network configuration wrapper over python deep learning library, Keras, that allowed us to configure different architectures. These different models were trained in-line on 375,000 tweets, validated (tuned) on 125,000 tweets, and then tested on 3000 tweets using cpu cycles from an i7 4790k Intel processor. It is important to note again, that this data was originally labeled with noisy emoticon labels by the sentiment140 team. Their labels, however, included classes outside of just positive and negative, as many emoticons represent neutral feeling as well. Following the approach of the sentiment140 team, we decided to train our models on only positive and negative tweets to better capture extreme polarity of tweet sentiment. Subsequently, we then alleviated all neutral testing examples from our dataset, thus reducing our output to two classes. This is a current limitation of our dataset.

During training, we collected intermediate accuracy scores for every 60 examples processed. Over trials of our different architectures, we used confusion matrix data to evaluate the precision and recall of our models on our test set. Below, in our results, we will show how our LSTM architectures were able to surpass expectations, leading us to reject the null hypothesis and accept the alternative hypothesis that says *LSTM network architectures do benefit the sentiment classification of tweets compared to traditional deep neural networks.*

RESULTS

1. **CONTROL - Traditional Deep Neural Net**

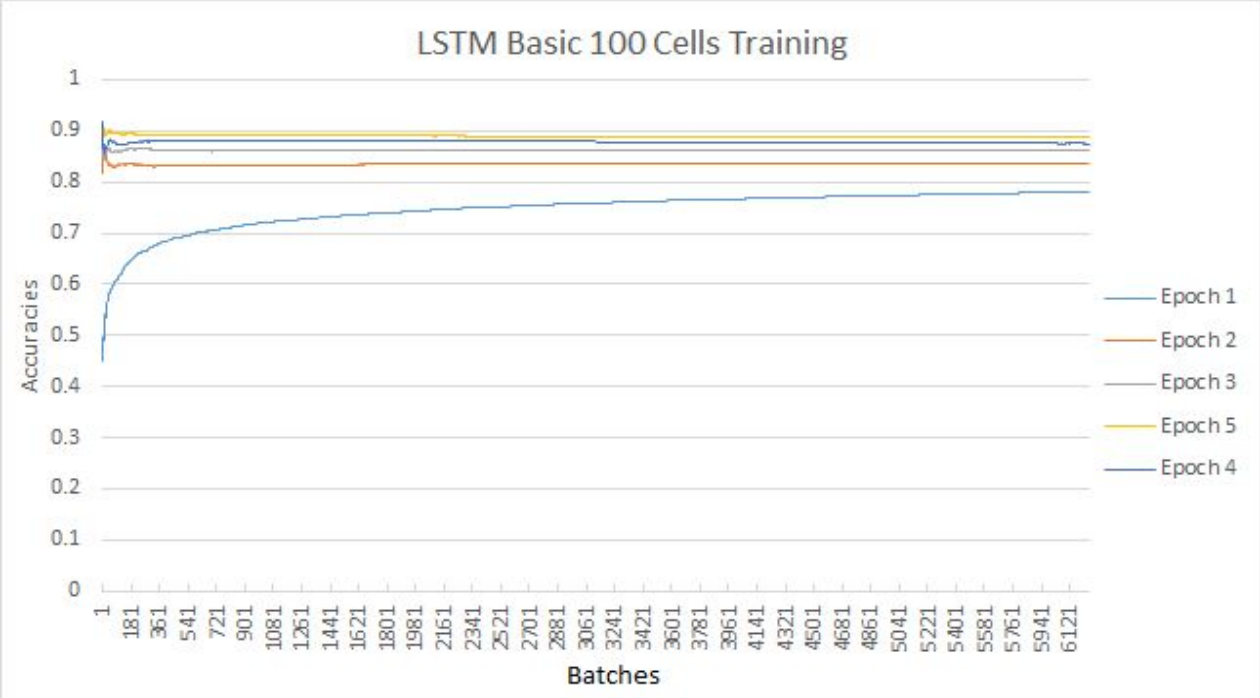
ARCHITECTURE	**TEST ACCURACY**
Dense_1, 100 units ⇒ Dense_2, 100 units ⇒ Dense_3, 10 units ⇒ Dense_4, 1 unit	78.43%

2. **LSTM BASIC - No Preprocessing or Feature Reduction, Dropout**

ARCHITECTURE	DROPOUT	TEST ACCURACY
1 LSTM Layer, 100 cells ⇒ Dense Layer, 1 unit	20%	81.45%
1 LSTM Layer, 50 cells ⇒ Dense Layer, 1 unit	20%	81.80%

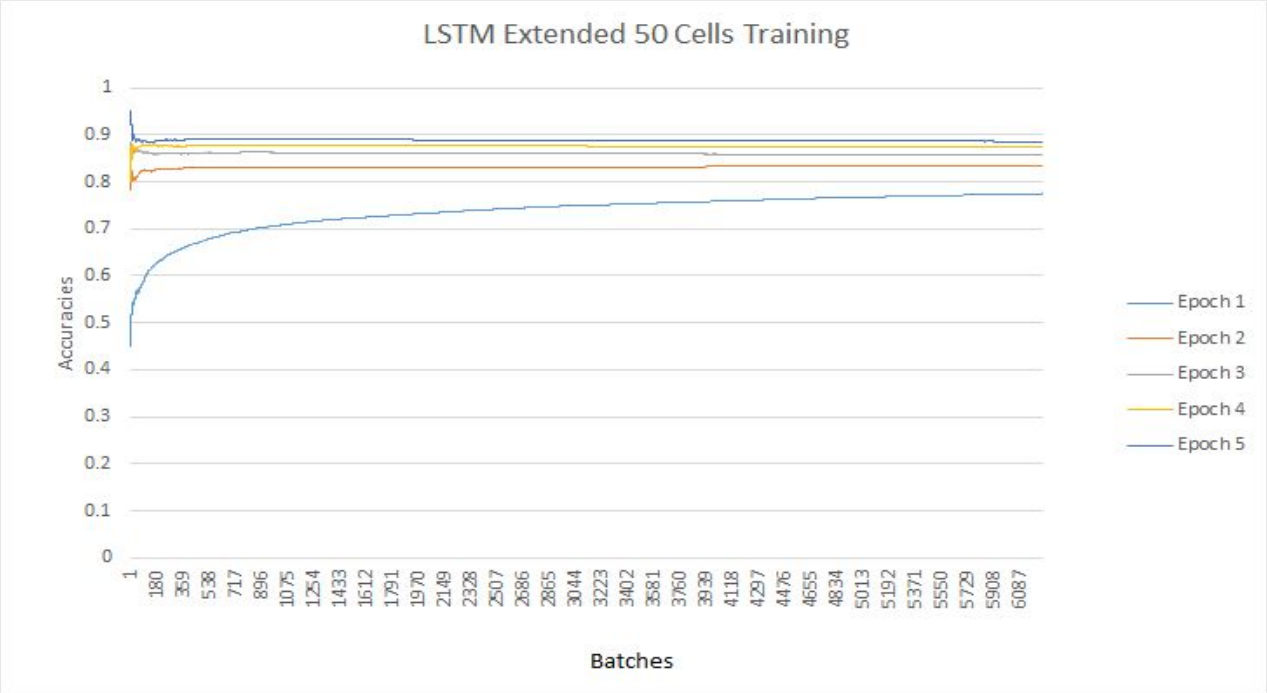
3. LSTM BASIC - Preprocessing and Feature Reduction, Dropout

ARCHITECTURE	DROPOUT	TEST ACCURACY
1 LSTM Layer, 100 cells ⇒ Dense Layer, 1 unit	20% LSTM	83.20%
1 LSTM Layer, 50 cells ⇒ Dense Layer, 1 unit	20% LSTM	83.03%
1 LSTM Layer, 1 cells ⇒ Dense Layer, 1 unit	20% LSTM	79.17%



4. LSTM EXTENDED - Preprocessing and Feature Reduction, Dropout

ARCHITECTURE	DROPOUT	TEST ACCURACY
LSTM_1, 50 cells ⇒ LSTM_2, 25 cells, ⇒ Dense Layer, 1 unit	20% LSTM	83.20%
1 LSTM Layer, 50 cells ⇒ Dense_1, 10 units ⇒ Dense_2, 1 unit	20% LSTM	83.03%



DISCUSSION

Our two highest scoring model results show rivaling accuracies. Our basic architecture model with 100 LSTM cells in one layer tied the accuracy of an extended model architecture with two LSTM layers, one with 50 cells, and the other with 25 cells. These models both had accuracies of 83.20%.

It is also worth noticing the substantial effect that preprocessing and feature reduction had on our models. We see that with preprocessing on the same model architecture, we gain close to 2% accuracy across model architecture.

From our results, it is easy to see the slight increase in accuracy performance with LSTM networks as opposed to the traditional, fully connected neural network approach. We were surprised to see, however, that a traditional approach still

performed adequately (79.43%) compared to the sentiment140 team's results of 82.9% accuracy with unigram Support Vector Machine models.

We decided to reject our null hypothesis and accept our alternative hypothesis that LSTM networks do in fact improve the accuracy of sentiment classification based on our differences in model accuracies. However, with more time, we would wish to assert this claim against more control or traditional models, as we only use one traditional, fully connected model architecture.

RELATED WORK

Due to its high popularity, there has already been a large amount of prior research in the topic of sentiment analysis. More importantly, there has been a large focus on sentiment analysis in the domain of reviews. A fairly up-to-date analysis of previous work in the topic of sentiment analysis up to this point can be found in a paper by Pang and Lee [9]. They studied the performance of other machine learning techniques for sentiment analysis, such as Naive Bayes, maximum entropy, as well as support vector machines, on movie reviews. Another interesting paper was by Basnal et al [10]. They researched the benefits of character level sentiment analysis versus word level sentiment analysis on tweets. They were interested in how LSTM networks would perform given in input of characters as opposed to words.

FUTURE WORK

Although our network performed well with the transformed dataset provided by Huang et al [2], we aim to explore other classification models. Using purely negative and positive labels for sentiment can be useful for general attitudes towards a particular subject, however, it does not do well in providing a further refined insight on said subject. Moving forward, we would like to introduce more possible classifications that could further refine our networks analysis. Our first step would be to explore further the idea of neutral tweets and begin to integrate them into our networks dataset.

Once our network can accurately distinguish the difference between negative, neutral, and positive sentiments in tweets, we plan on implementing an array of outputs to better represent the spectrum of emotion. Accurate classifications with this model can possibly help provide better insight as to why a general sentiment about a subject is viewed as positively or negatively. For example, a concert might seem to have a negative sentiment associated with it, however, with a more refined output on tweets regarding the concert, it may be evident that there are positive tweets regarding the performance, however there could be an overwhelming amount of negative tweets in regards to the bathrooms at the concert.

CONCLUSION

In our paper, we show that networks with the LSTM architecture perform slightly better in sentiment analysis in the domain of tweets due to ability to detect long term dependencies better than traditional deep learning networks. Text data like tweets is becoming increasingly more important as microblogging popularity rises and automated sentiment analysis is necessary to combat the ever growing amount of said data. As our project includes comparing LSTM networks against traditional deep neural networks, we hope that others can build upon the LSTM architecture to achieve more robust sentiment analysis on tweets.

CONTRIBUTIONS

This project was a group effort throughout the semester comprised of both individual tasks as well as group discussions. The initial proposal and conception of our network was brought upon through collaboration as well as our second and final approach. Once we had decided on an approach to proceed with, we began to divide up the tasks. To avoid unnecessary confusion and to reduce time when developing, Garritt Moede implemented the backbone structure that we would use throughout the rest of the research due to his previous knowledge with the frameworks we were to use. After our network was functional, we took the over-the-shoulder pair programming technique to debug and improve our network's performance. Soon we were able to divide up our different experimental runs to perform on our separate machines. The group again analyzed the data together and began to draw conclusions from our results.

SOURCES

[1] https://twitter.com/

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

[3] Bray, P. (2012, November 12). When Is My Tweet's Prime of Life? (A brief statistical interlude.). Retrieved May 9, 2017, from https://moz.com/blog/when-is-my-tweets-prime-of-life

[4] 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.

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

[6] https://about.twitter.com/company

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

[8] https://code.google.com/archive/p/word2vec/

[9] B. Pang, L. Lee, And S. Vaithyanathan. *Thumbs up? Sentiment classification using machine Learning Techniques*. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP), pages 79-86, 2002

[10] Trapit Bansal, Kate Silverstein, Jun Wang. *Character LSTM for Sentiment Analysis on Twitter.*