Text Sarcasm Prediction using Bi-Directional LSTM

Ang Kah Shin

Diploma in Applied AI &
Analytics, School of Computing
Singapore polytechnic.

***Abstract*— Neural Network-based methods have come a long way in the fields of Natural Language Processing. Classical Machine Learning algorithms were simply not powerful and efficient enough for the task of NLP. LSTM is a recurrent neural network that is capable of learning long-term dependencies between words. I propose to implement another form of LSTM, bi-directional LSTM layers, in my classification to allow for better retrieval of the past and future context of the sentence.**

I. INTRODUCTION

Sarcasm has been used by people to communicate the opposite of their true intentions in a humorous way to mock or tease others. More and more users are writing hurtful comments indirectly through the use of sarcasm to hurt/attack other users online. With the recent advances in Machine Learning, social media platforms have implemented different methods such as sentiment analysis, text processing to sieve out and filter as many toxic comments and hurtful content as much as possible. With the recent advances in the fields of Natural Language Processing, Long Short term memory models(LSTMs) have gotten much more accurate at correctly sifting out the intents of a given paragraph of text. In this technical paper, I will demonstrate an implementation of using bi-directional LSTM layers to further improve on retrieving the overall past and future context of the sentence to make better classifications of text sarcasm.

II. RELATED WORKS

In this Technical paper, I am looking out for related works that publish on

1) Early solutions to Text Classifications
2) Simple RNN solutions
3) LSTM methods

A. Early text Classification: A Naïve solution

In this paper, the researchers assume that a text is assumed to be processed sequentially, starting at the beginning of the document and reading input words one by one. The paper explores the suitability of one of the most popular methods for text classification, Naïve Bayes, to evaluate the capabilities of terms from documents. The researchers conducted an analysis on regular text classification tasks as well as sexual predator detection. The paper highlights that one of the most effective implementations is based on the multinomial distribution; where documents are represented by their term-frequency representation. However, I believe that measuring and classifying the words in a paragraph of text alone is not enough to properly estimate the importance and the weight of its meaning in the text. The paper mentioned that they were able to formulate out predictions for the input document regardless of the information inside.

B. Recurrent Neural Network for Text Classification with Multi-Task learning

The Recurrent Neural Network model in the paper was primarily used for mapping the input words and texts into dense vector representations to be combined with other forms of neural network architectures for classification. In the paper, the researchers mentioned that the modeling sequence was typically to feed the vector into a fixed-size vector using RNN and generate a probability based on the softmax output layer for classification. RNNs were able to perform much better than the regular Naive Bayes algorithm given that its forward propagation is computed with a memory value from the previous vectors. Unfortunately, for text sequences that were far larger and complex, the RNN layers were observed to have the problem of exponential gradient decay or growth. This problem with the 'exploding' or 'vanishing' gradients made it really difficult for the RNN model to retain correlations in long sequences.

C. LSTMs for Text Classification in Python

Long Short Term Memory (LSTM) is a type of Recurrent Neural Network that performs much better in terms of memory as compared to traditional Recurrent Networks. LSTMs consist of 3 Main gates:

1) FORGET Gate
2) INPUT Gate
3) OUTPUT Gate

The gates are responsible for deciding which information is kept, which information is important well as deciding what is the next hidden state of the value. These gates are also responsible for managing the internal memory keeping relevant memories while "forgetting" irrelevant ones. Because of the ability for LSTMs to "forget" information, it is able to efficiently improve its performance by only memorizing important information and does not suffer from the vanishing gradient problem. Given that LSTMs are able to effectively retain important information as well as manage information that might not be too relevant, it becomes optimal for use in text classification as it is able to use multiple word strings to predict "meanings/context" of the input string and allow for a much better interpretation of the input text for classification.

III. DATASET

In this paper, I have obtained a dataset from Kaggle which contains 28,619 rows of News Headlines, sarcasm labels as well as article links. The dataset contained 116 duplicated headlines.

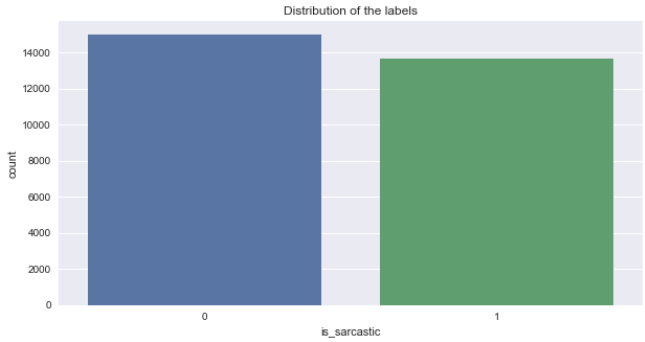

Fig 1.1 Class Distribution

Looking at the distribution of the labels, there appears to be slightly more non-sarcastic data (14985 rows) as compared to sarcastic data (13634 rows).

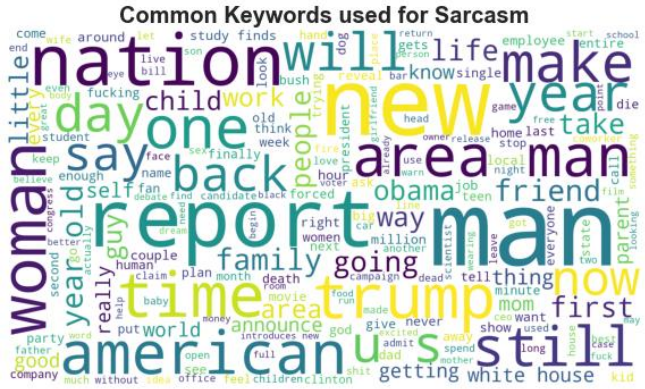

Fig 1.2 Common Keywords used for Sarcasm

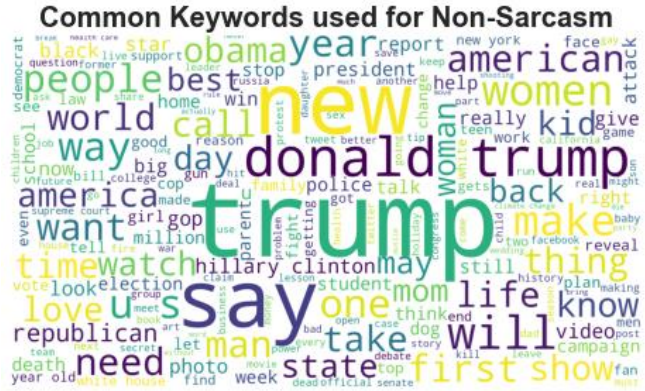

Fig 1.3 Common Keywords used for non-Sarcastic texts

I conducted further analysis to better visualize the common texts found in typical sarcastic headlines and non-sarcastic headlines.

IV. DATA PREPROCESSING

My first step was to remove any stopwords that are present in the news headline. the stopwords list used was obtained from the python NLTK stopwords built-in lists of stopwords. While removing the stopwords, we also dropped the article link column as there is no need for the article link to be present for the modeling as it does not relate to the features at all. The next step of the pre-processing was to convert all the alphabets in the headlines to the lowercase alphabet for standardization. I split the dataset into training and testing datasets of shapes (22895,) and (5724,) respectively. Since the model is not able to train on strings, I initialized a Tokenizer and fitted the tokenizer with the training dataset, specifying the *num_words* parameter to have a maximum length of 32.

```
array(['john lennons journey feminism matters era trump',
       'cleveland ohio magical place',
       'disapproving michelle obama printed fast food containers', ...,
       'protect inventors take trolls patent reform senator john cornyn ceo innovestion rackspace',
       'band dreams one day becoming popular enough alienate early fans',
       'bizarre plot blow target stores tank company stock'], dtype=object)
```

Fig 2.1 training data before tokenizing

```
array([[   74, 13590,  1473, ...,     0,     0,     0],
       [ 2633,  1233,  1181, ...,     0,     0,     0],
       [13591,   685,    19, ...,     0,     0,     0],
       ...,
       [ 1080, 26559,    73, ...,     0,     0,     0],
       [  850,   871,     6, ...,     0,     0,     0],
       [ 1671,  2386,  2177, ...,     0,     0,     0]], dtype=int32)
```

Fig 2.2 training data after tokenizing and padding

I fed the tokenized data through a series of pad_sequencing where the function will add paddings of '0' to the end of the tokenized array of the text so that every single row of text will have the same length when passed into the model.

V. EXPERIMENTATION

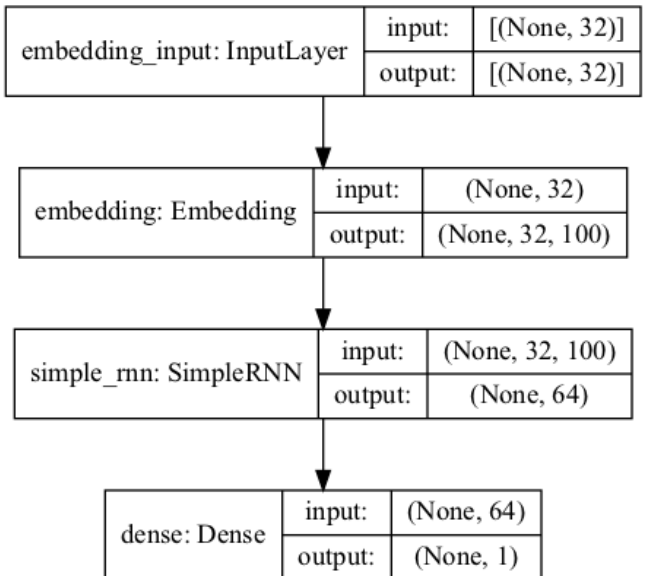

Fig 3.1 Model 01: Simple RNN

I first implemented a simple RNN model with 2872525 parameters to train. After 20 epochs of training with a batch size of 1024, I was able to achieve a training accuracy of 97.18% and a validation accuracy of 76.94%.

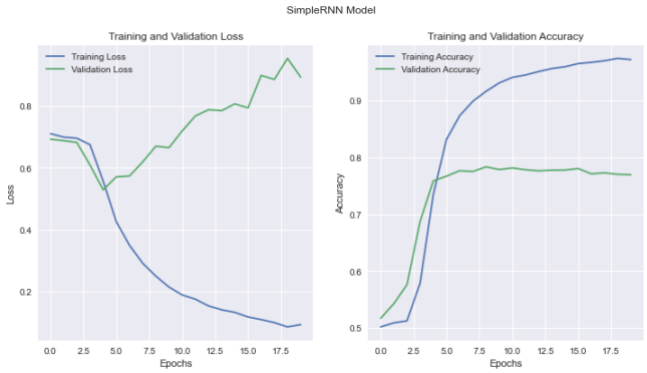

Fig 3.2 Loss-Accuracy plot for Simple RNN model

As you can see from the accuracy and loss plots of the training and validation, the training and validation loss is deviating and there appears to be a large variance in training and validation accuracy. This shows that the model is severely overfitting.

Secondly, I implemented a second model using 3 layers of LSTMs with a 0.4 dropout layer in between the second and third layers of LSTMs.

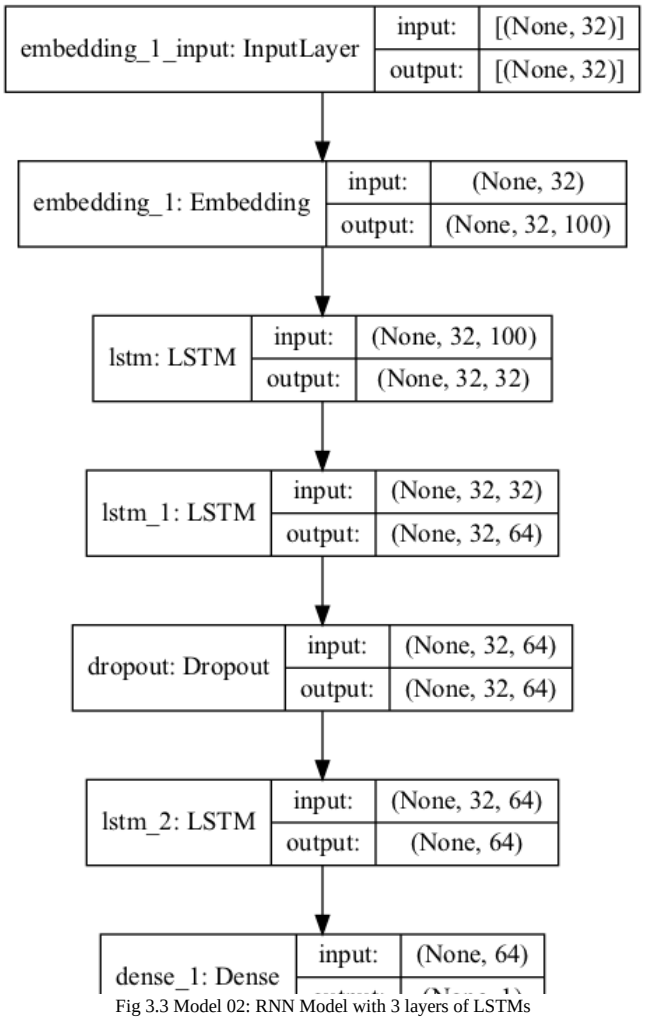

Fig 3.3 Model 02: RNN Model with 3 layers of LSTMs

The second model was trained with the same number of epochs (20), batch size of 1024, and the same training and validation data.

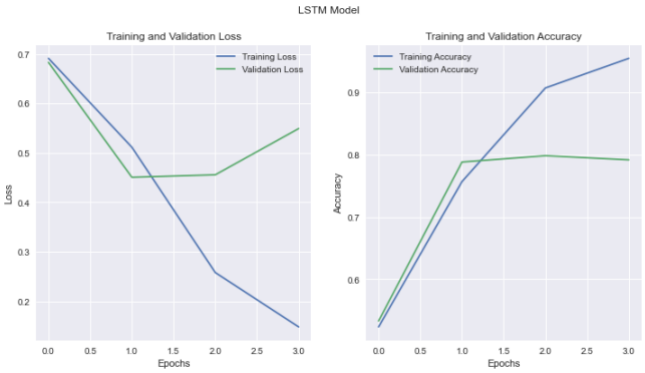

Fig 3.4 Loss Accuracy plot for LSTM-RNN model

The accuracy of the second model with the LSTM layers proved to be an improvement over the simple RNN model implemented before. This can be attributed to the strength of LSTMs being able to retain information over long text sequences (longest text sequence of 721 characters). However, the model is far from being accurate as the training and validation accuracy appears to be deviating.

Lastly, I implemented a third model utilizing the bi-directio nal LSTM layers with a dropout of 0.2 between the last and second last layers of LSTMs.

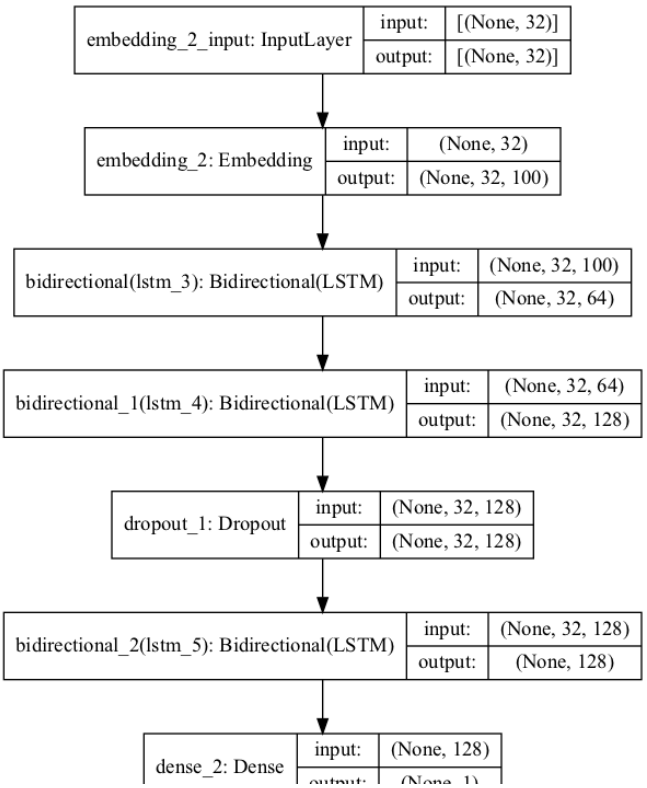

Fig 3.5 Model 03: bi-directional LSTM model

This model managed to yield a more stable and improved accuracy of 79 to 80% validation accuracy.

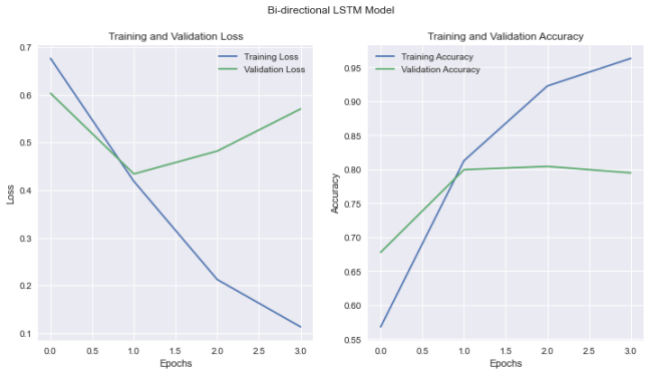

Fig 3.6 Loss Accuracy plot for bi-directional LSTM model

Although the accuracy and loss still appear to be deviating, the model appears to perform much better than the previous 2 models. This improvement in model performance is a small indicator of the amount of improvements a bi-directional LSTM that is able to understand and retain context of texts better.

VI. CONCLUSION

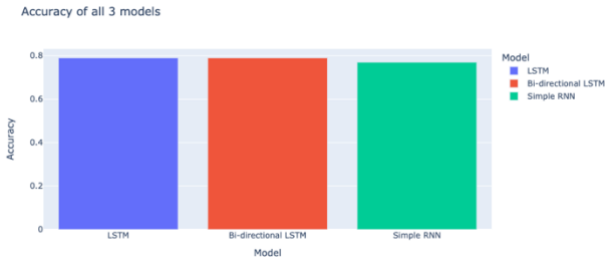

Fig 4.0 Model Accuracy

After experimenting with all 3 models, they all yielded very similar validation accuracy with the simple RNN model being slightly lower (77%) and both the LSTM and bi-directional LSTM models yielding accuracies of 79%.

Although the models are vastly underperforming and the differences shown here are very mediocre, I believe that with a much better dataset and implementation of the experimentation, the strengths of bi-directional LSTMs will outshine the other 2 networks; improving on the ability for the model to better classify texts.

In this technical paper, I covered on the usage and strengths of utilizing the ability of LSTM networks to "remember/forget" information for text classification. Similar concepts can be used on online chat rooms, and social media platforms to improve on the detection and filtering of toxic comments as well as online provocative comments.

REFERENCES

[1] Jair Escalante, H., Montes†, M., Villase˜nor†, L. and L. Errecalde, M., 2015. *Early text classification: a Naive solution*. [online] https://arxiv.org/pdf/1509.06053.pdf. Available at: <https://arxiv.org/pdf/1509.06053.pdf> [Accessed 26 November 2021].

[2] Liu, P., Qiu, X. and Huang, X., 2021. *Recurrent Neural Network for Text Classification with Multi-Task Learning.* [online] Ijcai.org. Available at: <https://www.ijcai.org/Proceedings/16/Papers/408.pdf> [Accessed 26 November 2021].

[3] Analytics Vidhya. 2021. *LSTM for Text Classification | Beginners Guide to Text Classification*. [online] Available at: <https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/> [Accessed 26 November 2021].