INDIAN STATISTICAL INSTITUTE, KOLKATA

Sarcasm Detection using Deep Learning

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT
FOR THE AWARD OF THE DEGREE

MASTER OF TECHNOLOGY
IN
COMPUTER SCIENCE

BY

Amit Kumar Jaiswar (CS1932)

UNDER THE GUIDANCE OF
Dr. Debapriyo Majumdar
Assistant Professor
Computer Vision and Pattern Recognition Unit, Kolkata
Computer and Communication Sciences Division Kolkata
Headquarters

CERTIFICATE

This is to certify that the dissertation entitled **"Sarcasm Detection using Deep Learning"** submitted by **Amit Kumar Jaiswar** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Debapriyo Majumdar
Assistant Professor,
Computer Vision and Pattern Recognition Unit, Kolkata,
Indian Statistical Institute,
Kolkata-700108, India.

ACKNOWLEDGEMENTS

I would like to show my highest gratitude to my advisor, *Asst. Prof. Debapriyo Majumdar*, Computer Vision and Pattern Recognition Unit, Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has literally taught me how to do good research, and motivated me with great insights and innovative ideas.

I would also like to thank to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

I would like to express my gratitude towards my parents and family member for their everlasting supports. I would like to support all of my friends for their help and support. Lastly, I also thank to all those, whom I have missed out from the above list.

Amit Kumar Jaiswar
Indian Statistical Institute
Kolkata - 700108, India.

ABSTRACT

Sentiment analysis is often used in NLP to understand people's subjective opinions. However, the analysis results may be biased if people use sarcasm in their statements. In order to correctly understand people's true intentions, being able to detect sarcasm is critical. Detection of sarcasm is the crucial step in sentiment analysis due to the inherent ambiguous nature of sarcasm. It also involves the complexity of language which makes sarcasm detection much harder, even for humans too. Therefore, sarcasm detection has gained importance in many Natural Language Processing applications. So, any progress in sarcasm detection, is a positive step towards pushing the boundaries of Natural Language Processing.

Many previous models have been developed to detect sarcasm based on the utterances in isolation, meaning only the reply text itself. Several researches have used both context and reply texts to detect sarcasm in reply and showed great improvement in performance. BERT, short for Bidirectional Encoder Representations from Transformers has been used for this specific task of sarcasm detection before and it has shown surprisingly good results for this task.

So, in this project, we aim to improve the accuracy of sarcasm detection by further exploring the role of contextual information in detecting sarcasm. We will also investigate the performances of different BERT based models where we have utilized the BERT embedding with different networks and compare the models for different data setups.

Contents

Chapter 1

Introduction

1.1 Introduction

Sarcasm, a sharp and ironic utterance designed to cut or to cause pain, is often used to express strong emotions, such as contempt, mockery or bitterness. Sarcasm detection play a key role in understanding people's true sentiments and opinions. Application of sarcasm detection can benefit many areas of interest of NLP applications, including marketing research, opinion mining and information categorization. However, sarcasm detection is a very difficult task, since it largely dependent on context, prior knowledge and the tone in which the sentence was spoken or written. Most sarcasm detection models so far consider the utterance in isolation. We investigated the role of conversational context information in sarcasm detection. More specifically, we explored how state-of-the-art models can make use of such context information for sarcasm detection and improve the results.

In this project, we followed the models explained by [Kumar and Anand, 2020] and adapted the off-the-shelve BERT-based classifier model from TensorFlow-Hub and ran the model on the available datasets. We have also paid particular attention to hyperparameters tuning and understanding the behaviour of the models. We further conducted a qualitative analysis on the effect of learning rate, number of hidden states and total number of epochs on sarcasm detection performance.

1.2 Importance-Research Perspective

- Much like QA, text summarization, machine translation, sarcasm detection involves complexity of language and is believed to be a much harder task
- Any progress in sarcasm detection, is an positive step towards pushing the boundaries of Natural Language Processing
- With improvement in understanding and approaches to sentiment analysis peoples are focusing on cases like aspect based sentiment analysis & sarcasm detection

So just considering the research perspective, it is worth investing time and energy in sarcasm detection.

1.3 What Makes Sarcasm Detection Difficult?

- It is deliberate people employ play of language
- It is subtle it is just a word, phrases or a punctuation that is here and there
- Sarcasm in text comes with challenges spelling mistakes, acronyms, slang words etc
- Even humans can find it hard to understand

1.4 Problem Statement

For given an unlabeled text T, build a solution such that it is able to detect whether T is sarcastic or not. We are using Reddit data and Twitter data.
This is a binary classification problem since one text can belong to either one of the classes.
There are two different classes :
i)Sarcastic
ii)Non-Sarcastic

Chapter 2

Related Work

This project is inspired by [Kumar and Anand, 2020], paper 'Transformers on Sarcasm Detection with Context', who have explored the role of conversational context and proposed various Transformers and LSTM based architecture for sarcasm detection. One challenge of sarcasm is the frequent necessity of prior knowledge.

Consider an example:

Context: What you love on weekends?
Response: I love going to the doctor.

Just by looking at the 'Response' we can tag this as 'Non-Sarcastic', but imagine this sentence as a reply to the 'Context', now we would to tag this as 'Sarcastic'. Hence it is necessary to know the context of a sentence to know sarcasm.

They have investigated the performances of different models (various LSTM and BERT models) and compare the models using response-only, context-only and context+response on different data setups.

Prior to this paper, [Ghosh et al., 2017] has also explored on the topic of sarcasm detection quite a bit, including his previous publication at EMNLP [Ghosh et al., 2015], which serves as the baseline for his paper in 2017 [Ghosh and Veale, 2017], it also contains more mature results and more deep learning model applications. In [Gregory et al., 2020] paper they have also worked on transformer-based approach to contextual sarcasm detection for Twitter dataset. In [Ilić et al., 2018] paper to capture complex morpho-syntactic features, they have proposed a model that uses character level vector representations of words, based on ELMO. In [Yang et al., 2016] paper they have discusses the concept of hierarchical attention network and applies attention on both word-level and sentence-level.

Our baseline model stems from the work of [Ghosh et al., 2018] about sarcasm analysis using conversation context. This model involves one LSTM reading the context and another reading the response while [Ghosh and Veale, 2017] also have proposed a similar architecture based on bidirectional LSTM to detect sarcasm in Twitter.

Lastly, we have also experimented with pre-trained transformer models [Devlin et al., 2018], for adaptation and training processes and its state-of-the-art high-performing results which has been very successful for other applications such as sentiment analysis, question answering, and recently even for sarcasm detection.

Chapter 3

Approach

Our approach to this project is to setting up and understanding a baseline model and implementing our own model on a larger dataset. These phases served the functions of allowing us to become acquainted with the models and dataset in a natural manner, and after that we also have tries to improve the performance of existing models.

For our binary classification task of sarcasm detection, we mainly implemented BERT from TensorFlow-Hub library and fine-tuned with our custom dataset. BERT is a multi-layer bidirectional transform encoder. Transforms are multi-head attention mechanisms (encoders and decoders) that are used in sequence dependency NLP tasks and have been proven to be more effective than Recurrent Neural Networks (RNN) and LSTMs. In short, Transformers attend to the problem in ways that RNNs cannot, due to their parallel nature. They look at all words in the window instantaneously, and learn importance and position in parallel. This meaning that words can "see" each other in ways impossible with an RNN, transformers are an especially useful tool when position and context for a word vectors are so important.

We began training the BERT implementation for our dataset, and run the training with the default hyperparameters. After trying many different configurations and after numerous unsuccessful attempts, we arrived at a working configuration of hyperparameters that we have kept the same for other models setups.

With the goal of extending an already impressive model into a more successful architecture, we tried to implement different models: one with adding a neural network with some dense layer on top of the BERT and another network with a combination of LSTM that utilize the compound sentiment scores of the input text. Lastly, we have also tried to utilize LSTM and Bidirectional LSTM networks over the BERT embedding for our sarcasm detection.

Chapter 4

Experiment

4.1 Datasets

There are two datasets that we have used in this project: Twitter and Reddit politics conversation threads.

Reddit Corpus [Khodak et al., 2018] collected 1.5 million sarcastic statement and many of non-sarcastic statement from Reddit. They self-annotated all of these manually and for our final dataset we have taken a subset of the corpus mostly containing the politics threads.
For Reddit dataset, each utterance contains the following fields:
1) **label:** Sarcasm label as 1 and Not Sarcasm as 0
2) **response**: the conversation response
3) **context:** the conversation context i.e., parent comment of the response.

Twitter Corpus [Ghosh et al., 2018] introduced a self-label twitter conversations corpus. The sarcastic tweets were collected by relying upon hashtags, like sarcasm, sarcastic, etc., that users assign to their sarcastic tweets and similarly for non-sarcastic tweets.
Unlike Reddit Corpus here we have only two fields:
1) **label:** Sarcasm as 1 and Not Sarcasm as 0
2) **response**: the conversation response as comment

We have taken the Twitter dataset size as same as the Reddit dataset and both the datasets are balanced between the 'Sarcastic' and 'Non-Sarcastic' classes and both of them have 31,172 posts. We have divided all data into train and test set with in ratio of 90:10 and we will be evaluating our model performance on these two dataset.

4.2 Data Pre-Processing

We used different text pre-processing technique on both the datasets to remove noise from text provided to us while keeping in mind that we also don't lose the useful information. We removed unwanted punctuation, multiple white spaces, hashtags, URL tags and links that were presented in each text. We also changed different contractions to their proper format, for example: "ol'" with "old" and "isn't" with "is not" etc.

4.3 Evaluation Method

For quantitative evaluation, we consider Precision, Recall and F1 scores.

Precision = $\frac{TruePositive}{TruePositive+FalsePositive}$

Recall = $\frac{TruePositive}{TruePositive+FalseNegative}$

F1 = $\frac{2*Precision*Recall}{Precision+Recall}$

We have used Precision, Recall and F1 scores as our primary metric to understand our model behaviour whether it is biased towards one label, and whether the selected train and test sets are balanced.

4.4 Baseline

We have implemented the LSTM Models as a baseline, for both the datasets utilizing one hot encoding and sarcasm indicators to perform two binary classification tasks. We have trained the model for response-only, context-only and context-and-response data.
The results for baseline model implementation are provided in the table below.

Dataset	**Model**	**Precision**	**Recall**	**F-1**
Reddit	$LSTM_R$	73.09	60.92	66.45
	$LSTM_C$	53.83	62.97	58.79
	$LSTM_{C+R}$	67.47	63.93	65.56
Twitter	$LSTM_R$	76.53	72.37	74.39

Table 4.1: Baseline results

4.5 Models

We implemented and fine-tuned BERT from TensorFlow-Hub library with our own hyperparameters and to pre-process plain text inputs into the BERT's standard data format we have utilized the pre-process model handler available on the TensorFlow-Hub.

We have used BERT base model (uncased) naming 'bert-base-uncased' pre-trained model in transformer layer. It has 12-layers, 768-hidden state size, 12-attention heads and 110M parameters, with each hidden state of (max-seq-len, 768) size and embedding output of 768 lengths. We have excessively used this BERT with different combination of networks on top of the embedding layer to achieve the better results.

Since, here we are performing binary classification so we decided to use Binary Cross-Entropy as the Loss Function and in order to reduce the loss we have used Adam with weight decay as the Optimizer with different learning rates in each setup and finally at the very last layer i.e., in classification layer we have used the Sigmoid as the Activation Function.

Few details are not common for all model setups like learning rates, number of epochs etc. So, we have listed them separately for all the model setups in the experiment details section.

4.5.1 BERT

This implementation has been done in TensorFlow, and for utilizing the BERT, we are importing it from TF-Hub library, we utilized this BERT and for pre-processing of the inputs we have used pre-process model handler from TensorFlow-Hub to make it BERT acceptable format.

These are the hyperparameters used:

- max-seq-length= 128, 256
- train-batch-size= 32
- num-train-epochs= 8, 9
- learning-rate= 2e-5 with decay
- loss function= Binary Cross-Entropy
- optimizer= AdamW

Fine-tuning follows the optimizer set-up from BERT pre-training. It uses the AdamW optimizer with a linear decay of a notional initial learning rate, prefixed with a linear warm-up phase over the first 10% of training steps. To avoid the overfitting of the model we have also used the early stopping monitoring over the validation loss.

We are considering it as the base model for BERT because we are just using the BERT embedding output and predicting the nature of sarcasm i.e., no network on top of BERT here.

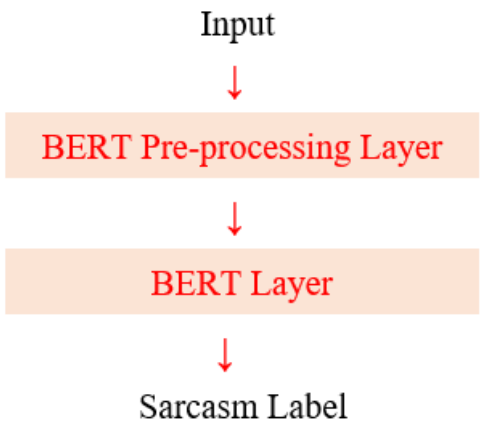

Figure 4.1: BERT Model

Training time for both the datasets it takes about 6 minutes for each epoch. This model yields better result with max-seq-length=128. So, in further model setups we are going to use this max-seq-length i.e., 128 only and from now on we are going to utilize the above BERT setup for other model the experiments.

4.5.2 BERT + Hidden Layer

We assumed that results may be improved by adding some dense layer as hidden layer on top of the BERT layer. In the base BERT model, there are 12 self attention layers whose output we were using for the classification. So here we have added 2-3 hidden layers and also used the dropout layers to prevent over-fitting of our model to the training data.

These are the hyperparameters used:

- max-seq-length= 128
- num-added-layers= 2, 3
- num-train-epochs= 7, 8, 9
- dropout-rate= 0.2, 0.3

Training time in this setup is about 6-7 minutes for each epoch for all the above combinations.

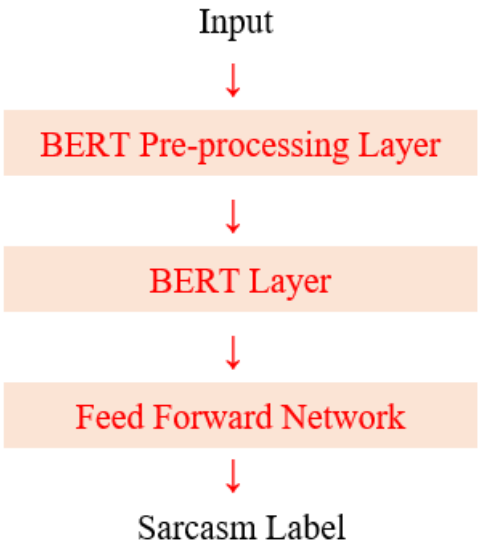

Figure 4.2: BERT Model with added hidden layer

4.5.3 BERT + Sentiment Score

We hypothesized that utilizing the compound sentiment scores of the sentences present in given text may help us to get better results. So, in this setup along with the BERT embedding output we have also concatenated the sentiment score of all the tokenized sentence that are present in text.

For the sentiment score we have used the VADER sentiment analyzer and before that for the tokenization we have use the NLTK's Sent tokenizer. We have implemented this setup with 1 hidden layer as well as without any hidden layers.

Here we have added a LSTM layer to process the VADER's sentiment score and the max number of tokenized sentence present in a text was 40, So we have made a vector for sentiment scores of the inputs with length 40.

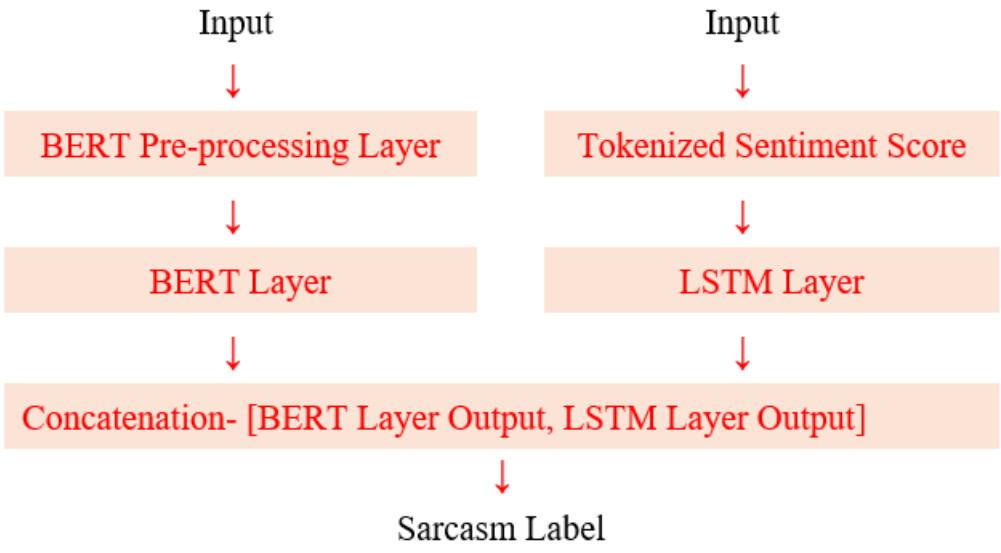

Figure 4.3: BERT Model with sentiment score

These are the hyperparameters used:

- max-seq-length= 128
- num-added-layers= 1
- num-train-epochs= 7, 8
- max-num-sentence= 40

Training time for this setup is also about 6-7 minutes for each epoch.

4.5.4 BERT + LSTM

In the same sequence of the experiment, we have also tried to add one LSTM layer on the top of the BERT layer and here we have used the last 4 hidden layer output of BERT as the input to LSTM so that it can capture the more contextual information and give better results. We trained this model with the initial parameters of BERT model.

These are the hyperparameters used:

- max-seq-length= 128
- num-added-layers= 1
- num-train-epochs= 8, 9
- BERT-hidden-layer-output-used = 4

Training time for this setup is also about 7-8 minutes for each epoch.

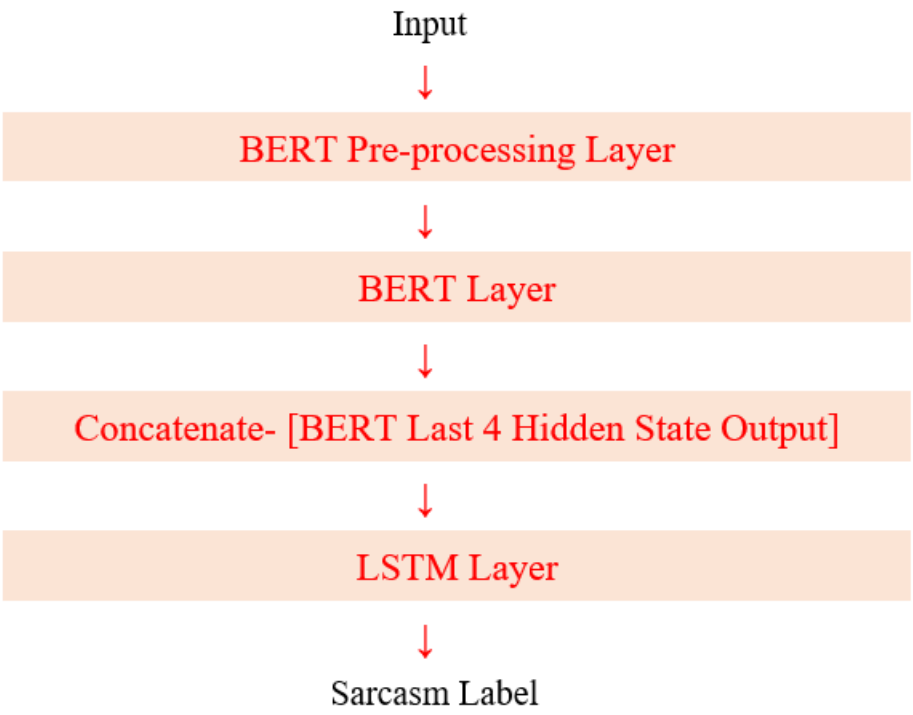

Figure 4.4: BERT Model with LSTM

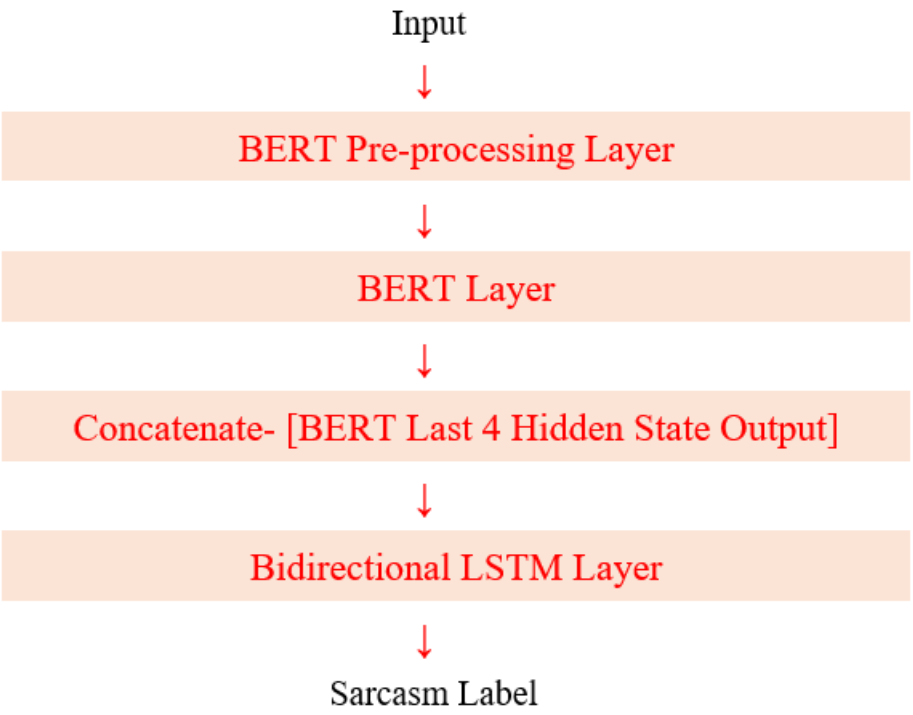

Figure 4.5: BERT Model with Bidirectional LSTM

4.5.5 BERT + Bidirectional LSTM

In this model setup we have replaced the unidirectional LSTM layer with the Bi-directional LSTM layer assuming that it will improve model performance with respect to the previous setup. We implemented this Bidirectional LSTM on top of the BERT layer keeping the all setups same as the previous model.

These are the hyperparameters used:

- max-seq-length= 128
- num-added-layers= 1
- num-train-epochs= 7, 8, 9
- BERT-hidden-layer-used = 4

Training time for this setup is also about 7-8 minutes for each epoch.

Chapter 5

Results

5.1 Experiment Results

The following table contains the summarized results for all the models that we have implement so far.

Dataset	Model	Precision	Recall	F-1
Reddit	BERT_R	75.58	74.27	74.92
	BERT_{C+R}	73.48	76.97	75.18
	BERT+Hidden Layer_R	73.26	**79.79**	**76.38**
	BERT+Hidden Layer_{C+R}	**77.21**	73.40	75.25
	BERT+Sentiment_R	74.23	73.40	73.98
	BERT+Sentiment_{C+R}	74.29	75.21	74.72
	BERT+LSTM_R	75.31	74.89	75.10
	BERT+LSTM_{C+R}	73.93	75.63	74.77
	BERT+Bi-LSTM_R	76.38	71.29	74.01
	BERT+Bi-LSTM_{C+R}	77.05	73.71	75.34
Twitter	BERT_R	**80.93**	84.56	82.70
	BERT+Hidden Layer_R	80.56	**85.26**	82.85
	BERT+Sentiment_R	79.57	84.88	82.14
	BERT+LSTM_R	82.85	83.60	83.22
	BERT+Bi-LSTM_R	82.96	84.61	**83.77**

Table 5.1: Model results

5.1.1 BERT-Baseline

Our baseline BERT model yielded to F1 score of 75.18 for Reddit test dataset and 82.70 for the Twitter test dataset and these results is better than other groups who used baseline BERT implementations for sarcasm detection. The scores are better than [Kumar and Anand, 2020] papers, which we have followed in this project. And the strong reason behind this could be that we have tried to implement it on a bigger dataset i.e. almost six times of the data that [Kumar and Anand, 2020] have used and it may be also because of the better BERT fine tuning.
These scores are higher than the baseline models, which is what we expected.

5.1.2 BERT + Hidden Layer

Our assumption of adding extra hidden layers on top of BERT layer turns out to be true and it has improved the prediction results by more than 1%. It seems like adding extra hidden layers have increased the ability of our model to understand sequence contexts in better way and that is why the prediction have improved.

5.1.3 BERT + Sentiment Score

We assumed that utilizing the sentiment scores of the sentences present in our text may help us to get better results. But it seems like compound sentiment score does not adding any extra dimension to BERT i.e., it may be capturing the the features that has been already learned by the BERT layer and also the possible reason could be the sentiment vector which is mostly sparse with 40 lengths.

5.1.4 BERT + LSTM

We were excited to see the results for LSTM models but we realized that a unidirectional LSTM is not giving us very robust predictions, as it wasn't learning contexts from both directions (start and end) in the sequence. Although its showing better results for Twitter dataset and for Reddit data its seems like the model a bit overfit.

5.1.5 BERT + Bidirectional LSTM

For the Reddit dataset our bidirectional LSTM implementation gave us scores close to the baseline, which was promising. But we were hoping that this will perform better. However, this model did not improve upon our unidirectional LSTM score. This was likely due to the fact that the Bi-LSTM overfit the training data and thus didn't improve performance on the test set. But for the Twitter dataset its behaviour is same like the previous model and performance has been improved for the Twitter dataset.

5.1.6 Result-Summary

These results have been obtained for the both datasets for our BERT classifier with response-only and concatenated context and reply with similar architecture of the model, such results offer great insights of the nature of the data sets and how sarcasm detection works for them.

Reddit: The Reddit dataset performs the best with response-only information. This is contradict our initial assumption from where we have started. However, it's possible that the Reddit response-only dataset contains many flags for sarcasm, which are also present in the large dataset of other NLP tasks that BERT is pre-trained with. Compared to the LSTM response-only performance, which also did fairly well, it's also likely that the Reddit reply dataset contains various outstanding marker for sarcasm which make it easy to identify based on response alone. And of course, one the possible reason could be that it has smaller sequence lengths.

Twitter: The Twitter data that we have only contains the responses and compared to the LSTM response-only performance, we can say that it also did fairly well on BERT models.

Chapter 6

Analysis

6.1 Nature of the Data

Overall, BERT has outperformed our baseline LSTM models with a significant margin, for both datasets. But it has different accuracy range for both the datasets. We suspect the main reason is due to the nature of these two data sources. Twitter has shorter sentences in each entry and all entries are labeled and selected by crowd-sourcing while on the other hand self-annotated Reddit data has never gone through professional human processing, the entries cover many different topics and both datasets contain internet slang.

6.2 Data Pre-Processing

Another thing we realized that there are several other pre-processing techniques that we can use to possibly improve the performance of our models. Due to the nature of these two datasets, the context and response texts often include word tokens that do not find perfect mappings in pre-trained word embeddings. This is mainly because of their own emojis or markdowns present in text. Usually, the usage of certain markdowns and emojis gives strong indication of sarcasm. However, in our pre-processing, these patterns are either being tokenized into different tokens or treated as words not found in the pre-trained embeddings.

6.3 Sequence Length vs Incorrect Predictions

We also investigated whether sequence length had any effect on incorrect predictions, or whether there are any patterns in between sequence length and correct predictions. And one thing we found that our BERT model gives better performance with sequence lengths less than 128 and as we have increased the max-seq-length to 256 the performance got dropped a bit. So, here we can say that it giving correct predictions for smaller sentences.

Chapter 7

Conclusion

In this project, we experimented with different types of network layer on top of BERT model for sarcasm detection with both response-only and context-and-response for Reddit data [Khodak et al., 2018] and for Twitter data [Ghosh et al., 2018] just for the response-only, and analyzed their performances. The results show that for BERT models with two hidden layer can actually improve model performance for Reddit data and in case of the Twitter it is best for BERT+Bidirectional LSTM. This model generates better results comparing to previous BERT+LSTM model developed in [Kumar and Anand, 2020] papers. More importantly, we believe that the pre-trained BERT classifier can achieve state-of-the-art results for sarcasm detection here we have tried with a bigger version of both the datasets and it outperforms our baseline model by almost 10% and these scores are better than other groups who have used baseline BERT implementations for sarcasm detection.

For future work, we plan to improve our current model, either by fine tuning or modifying our current LSTM based model or also adding with some other networks like CNN or the combination of both on top of the BERT, so that the network can have better understanding of sarcasm with context and it learns in better way that how sarcasm triggered.

Bibliography

[Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Ghosh and Veale, 2017] Ghosh, A. and Veale, T. (2017). Magnets for sarcasm: Making sarcasm detection timely, contextual and very personal. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 482–491.

[Ghosh et al., 2017] Ghosh, D., Fabbri, A. R., and Muresan, S. (2017). The role of conversation context for sarcasm detection in online interactions. *CoRR*, abs/1707.06226.

[Ghosh et al., 2018] Ghosh, D., Fabbri, A. R., and Muresan, S. (2018). Sarcasm analysis using conversation context. *Computational Linguistics*, 44(4):755–792.

[Ghosh et al., 2015] Ghosh, D., Guo, W., and Muresan, S. (2015). Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1003–1012.

[Gregory et al., 2020] Gregory, H., Li, S., Mohammadi, P., Tarn, N., Draelos, R., and Rudin, C. (2020). A transformer approach to contextual sarcasm detection in twitter. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 270–275.

[Ilić et al., 2018] Ilić, S., Marrese-Taylor, E., Balazs, J. A., and Matsuo, Y. (2018). Deep contextualized word representations for detecting sarcasm and irony. *arXiv preprint arXiv:1809.09795*.

[Khodak et al., 2018] Khodak, M., Saunshi, N., and Vodrahalli, K. (2018). A large self-annotated corpus for sarcasm. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

[Kumar and Anand, 2020] Kumar, A. and Anand, V. (2020). Transformers on sarcasm detection with context. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 88–92.

[Yang et al., 2016] Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.