[Team 32] Sarcasm Detection in News Headlines

Meenakshi Madugula(mmadugu), Roshani Narasimhan(rnarasi2), Sankalp Gaharwar(ssgaharw)

***Abstract*—Sarcasm detection within news headlines can serve as an important metric, not only for gauging how relevant a news article is, but also for indicating the quality of the news agency who publishes the article. We aim to identify and classify sarcastic news headlines. Sarcasm detection needs us to perform the task of encoding the prior context associated with the words, in tandem with the set of reference words. The prospect of using the historical states for classification lends itself brilliantly to the use of Neural Network architectures. Thus, we build a LSTM model and a transfer- learning-based BERT model for our Sarcasm Detection tasks. We explore how the associated parameters for each of the model need to be tuned for being able to perform the classification tasks with maximum accuracy. Finally, we empirically compare the performance for the best implementations of our models and suggest the most apt approach for the Sarcasm Detection task. The dataset is taken from Kaggle [4].**

I. Model Training and Selection

A. The Model

Long short-term memory(LSTM) models are widely popular in dealing with text-based data mining tasks as they are designed to exploit the inherent sequence present in textual data. LSTM which is a modified RNN, can also *remember* the context of the data [3]. LSTM were however developed with the goal of solving the vanishing gradient problem, which were predominant in traditional RNNs. An LSTM unit consists of a cell and three gates, namely - an input gate, an output gate and a forget gate. The cell retains values it has seen previously, for arbitrary intervals of time. Three gates are responsible for regulating the flow of information across the cell. All gates use the sigmoid activation. The input gate determines which value from input should be used to modify the cell's hidden state. The output gate uses the input and the memory of the unit to decide the output. The forget gate decides what information should be kept based on the information passed from the previous hidden state and the current input. The tanh activation helps in bringing the values provided to it, to the range of -1 to 1.

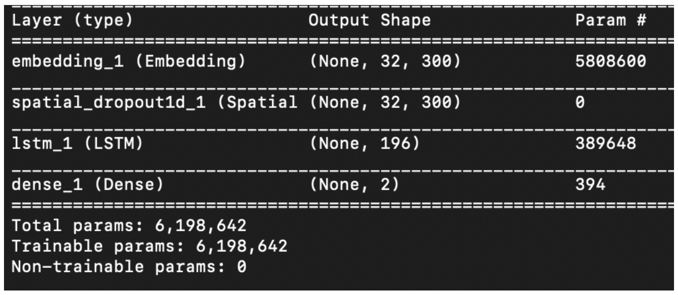
We believe the use of LSTM [7] model will facilitate us by retaining the context of the comments in memory, to determine whether a headline is sarcastic or not. Due to their property of being able to selectively remember patterns for long duration of time, LSTMs have an edge over conventional feed-forward neural networks and RNNs.

We used a tokenizer from Keras [1] to convert the data to a format that is suitable for the LSTM model. The tokenizer assigns a unique token for all the words present in the training dataset and converts every sentence to a sequence of numeric tokens. The maximum vocabulary of the tokenizer could be tuned to allow the model to train faster, however since we focused on getting better accuracy, we decided to not have a lower limit for the maximum number of words than the words present in the training dataset. The sequences generated by the tokenizer were same length as that of the sentences (one token per word). In order to make all the sequences of uniform length, we used zero padding and limited the length of each sequence to a maximum sequence length of 32. We decided on the value 32, since most headlines in the entire dataset consisted of less than 32 words.

We used the GloVe [6] embedding as the first layer of our LSTM model, to facilitate it to perform better. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. We used the pre-trained word vectors which were generated based on Wikipedia 2014 + Gigaword 5 corpora. There were three sets of word vectors of embedding dimension 100, 200 and 300 respectively, with larger dimensions capturing more information. Thus the embedding dimension is a hyper parameter - higher dimension of the vector for each word leads to capturing more information. We provide the tokens generated by the tokenizer as input this embedding layer, thus each input sequence is of length 32 and the vector corresponding to each token is generated based on GloVe. We allowed the pre-trained embedding layer also to learn during the training process.

We added a spatial dropout layer with a dropout value of 0.4 which helps in promoting independence between one dimensional feature maps instead of the individual elements. Then we added the LSTM layer with 196 units and tried different values of dropout, recurrent dropout (used for back propagation) and activation functions. The output dense layer is activated using a softmax function. The model summary is shown in Figure 1. The hyperparameters for an LSTM model are the number of units in the LSTM, number of epochs, activation function, learning rate, optimizer, dropout rate and recurrent dropout rate. We would be focussing on tuning these hyperparameters for our task.

Another model we use is BERT [2] - a transfer-learning based technique developed by Google, which has been pre-trained on Book Corpus and the Wikipedia database. It has been trained on two tasks - next word prediction and next sentence entailment. Since the BERT model is capable of understanding language, we hypothesize that it can grasp the inherent sarcasm of a headline efficiently. The fine tuned pre-trained BERT model uses transfer learning to predict if the sentence is sarcastic. We chose BERT base uncased model for our experiments. It has 12 layers with a hidden size of 768 per



```
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 32, 300)           5808600
_________________________________________________________________
spatial_dropout1d_1 (Spatial (None, 32, 300)           0
_________________________________________________________________
lstm_1 (LSTM)                (None, 196)               389648
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 394
=================================================================
Total params: 6,198,642
Trainable params: 6,198,642
Non-trainable params: 0
```

Fig. 1. LSTM Model Architecture

layer. BERT expects input in a specified format. The sentences are preceeded with a special [CLS] token. This token will be used to make the final prediction. The sentence is then padded by a [PAD] token to matain a uniform sequence length across sentences. The model architecture is shown in Figure - 2. The output from the [CLS] token shown as class label in the figure is passed through a dense softmax layer. The probabilities from the softmax layer are used to predict a 0 or 1. Hyper parameter tuning was performed to select the optimal sequence length of a sentence and the optimal number of epochs. Since BERT is computationally expensive, the fine tuning with our dataset takes about 5-6 hours to train for 3 epochs.

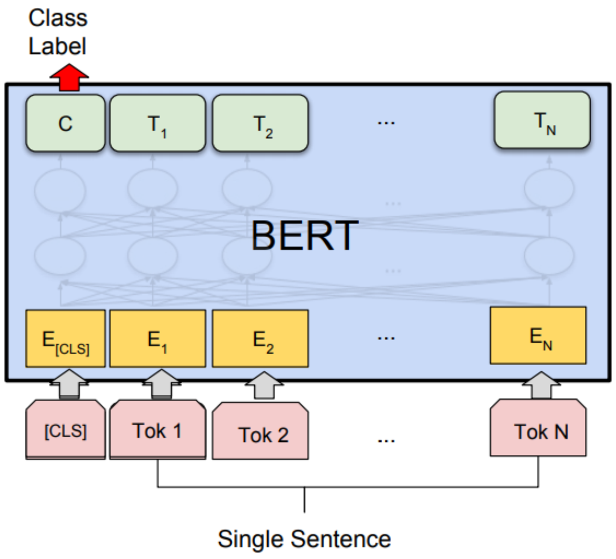


Fig. 2. BERT Model Architecture

B. Baseline

For our baseline model, we are using the Naive Bayes Classifier that is built using the concepts of the Bayes Theorem. The premise for choosing this model is that it allows us to map the posterior probability distribution for the context information data within our corpus. By building predictors that are independent of each other in terms of the features that they consider for classification. Naive Bayes helps us to build a Sarcasm Detection model that is not only quite accurate, but fairly speedy in terms of computation time as well. We implement a Gaussian Naive Bayes classifier that is built using a Normally-Distributed probability distribution for the features within our Sarcasm Detection data set. For building the model, we use the Sci-Kit Learn library within Python.

We find that the Naive Bayes classifier does not give us much options to perform hyper-parameter-tuning and hence does not perform classification tasks in a manner that is competitive with the Neural Network Architectures such as LSTM and Transfer-learning approaches such as BERT. Both BERT and LSTM give us ample scope for performing hyper-parameter tuning and thus, we are able to adapt our model's parameters for improving the classification accuracy. Among the two, we expect the BERT model to perform better than the LSTM model since it is much more adapted to Language-oriented classification tasks and hence, we can model our Sarcasm Detection classifiers on the features within our data set.

II. Experimental section

A. Metrics

Since our aim is to filter sarcastic comments, it is important that our model does not filter out legit comments. Hence the main focus is on the precision of (particularly label "1") rather than accuracy. To get a complete idea how the model performs, we supplement the precision metric with F1 score to evaluate the model. Additionally, metrics such as recall, accuracy and auc-roc score are reported along with the confusion matrix.

B. Model Selection

The Naive Bayes Model is basic and not hyper parameter tuned. The performance of the model was not satisfactory. Hence we decided to tune a LSTM model to be considered as a baseline.

We split the data into training plus validation and test data in the ratio 7:3. Train plus validation data was split in the ratio 7:3 to make the final training and validation data. We used the same datasets for our experiment with the BERT model as well.

The hyper parameters for the LSTM layer were dropout values, learning rate, optimizer, activation function and number of epochs. A separate validation dataset and grid search method was used to select the best parameters. The different optimizers tried were SGD, RMSprop and Adam. While using SGD, the loss and accuracy values did not change over the epochs. The model froze and resulted in vanishing gradients as there was no improvement over the epochs. RMSprop and Adam performed almost the same in terms of the validation accuracy. Since Adam internally uses RMSprop, Adam was selected. Learning rate values [1e-1,1e-2] were tried but the loss function was unstable and jumping around. Hence we decided to try [1e-3,1e-4], out of this 1e-3 was observed to perform well as the curve was smooth.

Activation functions ['tanh', 'relu', 'sigmoid'] and Dropout [0,0.2,0.5,0.7] were combined and tried. A few of the model architectures and their results are listed in the table I. The results listed are with an embedding layer, followed by 196 LSTM units and a softmax layer. If the validation and training accuracy are far apart it means that there is over fitting. From the table I we can see that the first model has a training accuracy of 0.99 but only 0.81 validation accuracy. Hence increasing the dropout values should help over come this. This is observed as a trend in the first 3 rows of the table where the activation is ’relu’ function as both the accuracy values move closer to each other. The next 3 rows of the table correspond to the ’sigmoid’ function. However there is overfitting again without dropout (Row 4 of the table). The same trend is followed here as well as the dropout values increases. It is also seen for the ’tanh’ function as well. Overall the ’sigmoid’ performs worse on the validation accuracy. But there is not much difference between the ’tanh’ and the ’relu’ function. However, the loss function curve of ’relu’ was found to be smoother than that of ’tanh’. Observing the dropout values from the table overfitting is slightly overcome with a dropout of 0.5 (’relu’) with train accuracy at 95 and validation accuracy at 82. The loss values are also close enough. The trends in the table are consistent to show that having a dropout helps the model to perform better.

To select the number of epochs, the learning curve for accuracy and loss was plotted for all the models. This was implemented using the history object returned by the Keras [1] fit method. The validation and training loss and accuracy were plotted against 20 epochs. The curves are shown in Figure - 3 and Figure - 4. From the graphs, there are signs of overfitting as both curves are farther apart. Hence we decided to increase the dropout to 0.7 and also decrease the dimensions of the embedding to 100. The performance of this model is shown in the last row of Table I. We can see that the train accuracy is 0.87 and validation accuracy is 0.81. The gap between them is further reduced. The corresponding learning curves are shown in Figure 5 and Figure 6. It can clearly been seen from the figures that the gap has been bridged between the training and validation accuracy as well as loss.The curves are found to be flat after the first few epochs, Hence 10 was selected as the optimal number of epochs. Hence the model with embedding layer of dimension 100, ’relu’ activation, 196 hidden units in LSTM layer and dropout of 0.7 was selected as the best model. This model also acted as the baseline for our language model.

Since the BERT [2] model is pre-trained, there were only a few parameters to tune. Since running BERT is computationally expensive we did only a few runs. The hyper parameters were sequence length, batch size, learning rate and number of epochs. The maximum sentence length for our data was around 34. Hence 32 was selected as the maximum sequence length. The batch size of 32, 64 and 128 was tested. The optimal results were found were of size 32. Learning rate was tried in the ranges [1e-3,1e-5,2e-5]. 2e-5 was found to be optimal. The maximum epochs tested were for 5, however the optimal number to avoid over fitting was found to be 3.

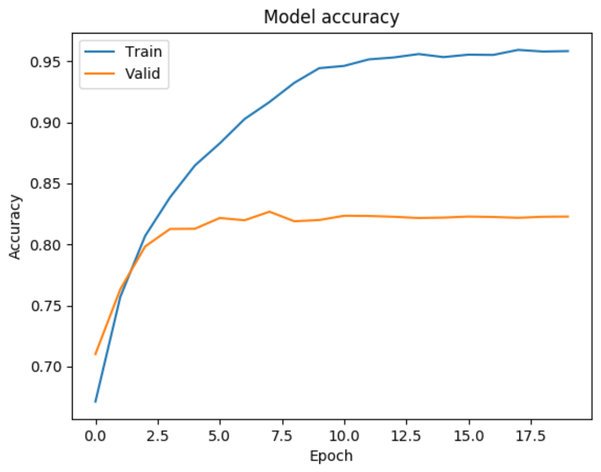


Fig. 3. Learning curve for accuracy with 300 Embedding & 0.5 dropout

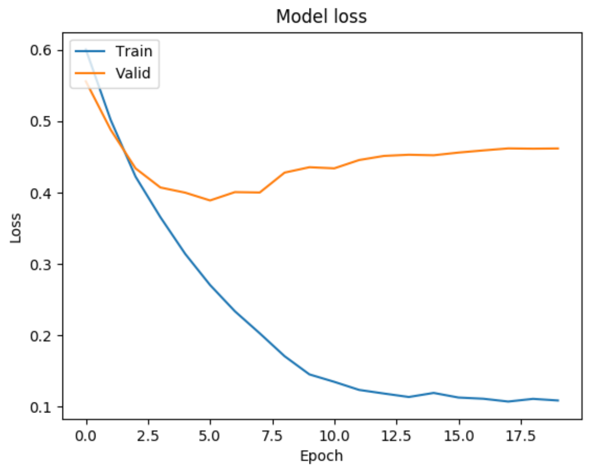


Fig. 4. Learning curve for loss with 300 Embedding & 0.5 dropout

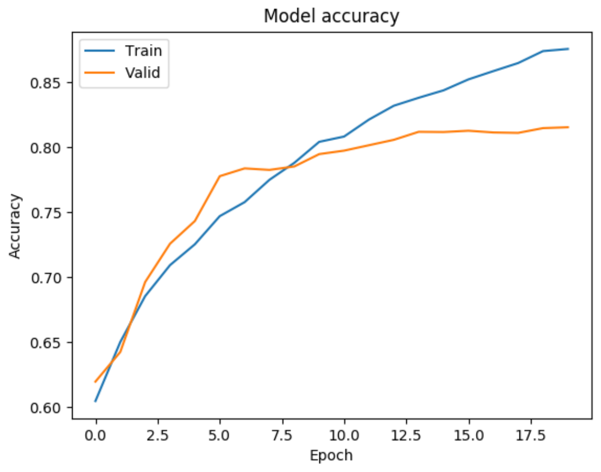


Fig. 5. Learning curve for accuracy with 100 Embedding & 0.7 dropout

TABLE I
MODELS PERFORMANCE COMPARISON

Dropout	Activation	Train Acc, Loss	Valid Acc, Loss
No	relu	0.99, 0.02	0.81, 0.75
0.2	relu	0.97, 0.06	0.82, 0.57
0.5	relu	0.95, 0.1	0.83, 0.46
No	sigmoid	0.98, 0.04	0.79, 0.73
0.2	sigmoid	0.97, 0.07	0.79, 0.64
0.5	sigmoid	0.94, 0.13	0.80, 0.52
No	tanh	0.99, 0.02	0.81, 0.77
0.2	tanh	0.97, 0.05	0.82, 0.64
0.5	tanh	0.94, 0.12	0.83, 0.53
0.7	relu (100 Embed Dim)	0.87, 0.28	0.81, 0.39

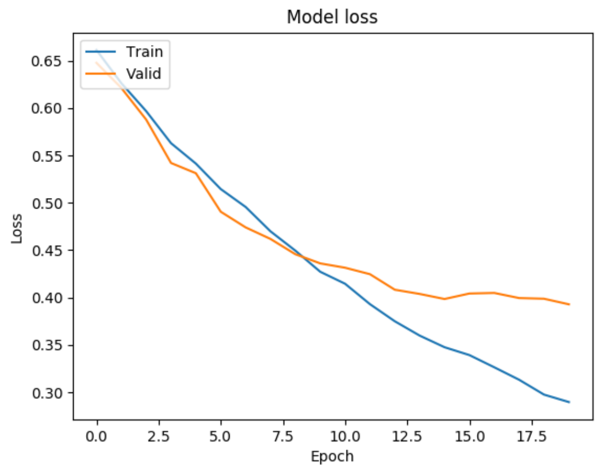


Fig. 6. Learning curve for loss with 100 Embedding & 0.7 dropout

C. Performance and Comparison to Baseline

The models were evaluated for Precision, Recall, F1 Score and Accuracy metrics. We also looked at the confusion matrix. The metrics of all the three models - Naive Bayes, LSTM (tuned as per previous section) and BERT are listed in Table - II, Table - III and Table IV respectively. Their corresponding Confusion matrices are shown in Table - V, Table - VI and Table VII respectively. We used the classification report and confusion matrix function of the sklearn package [5].

From Table II the precision of label 1 seems high but the F1 score for the label 1 is only 23%. Label 0 seems to perform well. From Table V many very few sentences were predicted as sarcastic, hence the precision is high but there is a hit in the recall and F1. Since this was one of the baseline models, let us compare it with the other models. There is a jump in precision, recall and F1 for label 1 from Naive Bayes to LSTM model. This can be observed from table II and III. Since sarcastic classification is a difficult problem that requires more than the probabilities of the words, Naive Bayes does not perform well. LSTM is able to classify sentences better by using its inherent properties. It archives a precision of 84% and F1 of 80% for label 1. The confusion matrix for LSTM (Table VI) show that only 435 samples were misclassified for label 1. BERT outperforms both the baseline models as it is a language model and is able to better understand sarcastic contexts. BERT archives a 95% precision for label 1 with a F1 of 92% (Table - IV). It has a significant increase in performance from the LSTM model. From the confusion matrix (Table VII) it can be seen that very few examples are misclassified compared to the LSTM and Naive Bayes model. Hence we can conclude that the pre-trained language model can boost the performance of sarcastic comment classification task.

TABLE II
NAIVE BAYES PERFORMANCE METRICS

Labels	Precision	Recall	F1 Score	Support
0	0.54	0.95	0.69	3113
1	0.71	0.14	0.23	2897
Macro Avg	0.63	0.54	0.46	6010
Weighted Avg	0.62	0.56	0.47	6010

TABLE III
LSTM PERFORMANCE METRICS

Labels	Precision	Recall	F1 Score	Support
0	0.80	0.86	0.83	3113
1	0.84	0.77	0.80	2897
Macro Avg	0.82	0.81	0.81	6010
Weighted Avg	0.82	0.82	0.81	6010

TABLE IV
BERT PERFORMANCE METRICS

Labels	Precision	Recall	F1 Score	Support
0	0.91	0.95	0.93	3113
1	0.95	0.89	0.92	2897
Macro Avg	0.93	0.92	0.92	6010
Weighted Avg	0.93	0.92	0.92	6010

TABLE V
NAIVE BAYES CONFUSION MATRIX

True—Pred	0	1
0	2955	158
1	2504	393

TABLE VI
LSTM CONFUSION MATRIX

True—Pred	0	1
0	2678	435
1	675	2222

REFERENCES

[1] F. Chollet et al. Keras. https://keras.io, 2015.
[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

TABLE VII
BERT CONFUSION MATRIX

True—Pred	0	1
0	2964	149
1	305	2592

[3] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[4] R. Misra and P. Arora. Sarcasm detection using hybrid neural network. *arXiv preprint arXiv:1908.07414*, 2019.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[6] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. pages 1532–1543, 2014.

[7] A. Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *“Elsevier ”Physica D: Nonlinear Phenomena” journal, Volume 404, March 2020: Special Issue on Machine Learning and Dynamical Systems”*, abs/1810.04805, 2018.