CS 410 Text Information Systems Course Project Final Report

**Team Name:**

Virginia DS

**Team Members:**

Huanzhen Hu NetID: [hh21@illinois.edu](mailto:hh21@illinois.edu) Role: Leader

Ying Zhang NetID: [ying12@illinois.edu](mailto:ying12@illinois.edu) Role: Member

**Topic Selected:**

Text Classification Competition

**Introduction**

For this Text Classification Competition project, we are trying to detect whether a Tweet message is sarcasm. There were 5000 training data was given and 1800 test data need to be predicted. For training data, label (SARCASM or NOT\_SARCASM), response (the Tweet to be classified), and context (the conversation contest of the response) were given. For test data, response, context, and id (string identifier for sample) were given.

To Complete the classification task, we tried the classic tokenization method and machine learning technique but failed to reach the baseline. Therefore, we moved forward to other cut-of-the-edge NLP techniques, including CNNs, LSTM, and BERT. Most of those techniques reached SOTA results in previous research. We conducted several experiments and compared different outputs. As a result, BERT-series model performed best among all methods we tried.

**Experiments**

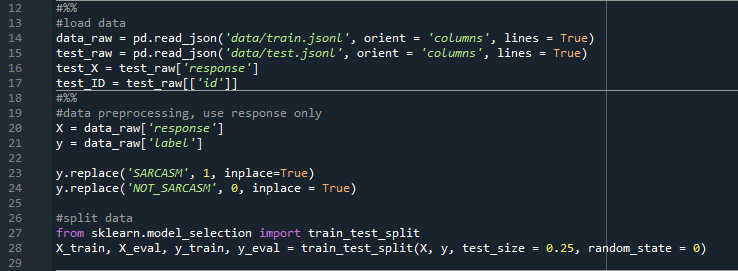
**Classic Tokenization & Machine Learning Techniques**

i. Data Preparation

The provided dataset for training consists of three parts: response, context, and label. Response is the tweet document to be classified; Context is the conversation context of the response; And label contains two categories – “SARCASM” or “NOT\_SARCASM”. To simplify the problem, we only consider the response as input “X” and the label as input “y”. For convenience, we replaced “SARCASM” with integer 1 which indicates positive and replaced “NOT\_SARCASM” with integer 0 which indicates negative.

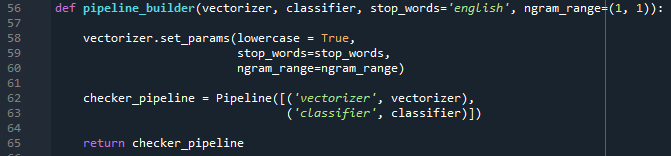
The provided dataset for testing consists of three parts: id, response, and context. To be consistent with the training data, we only utilized the response for classification.

Once the data was well-prepared, we randomly split training data into two parts – train and evaluation with a ratio of 3:1. Please check the code below for more details:

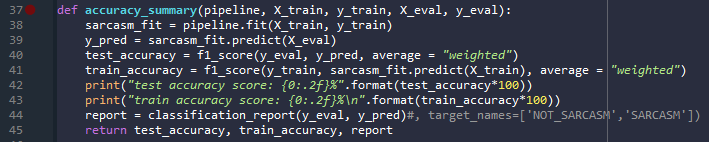


ii. Model Training and Prediction

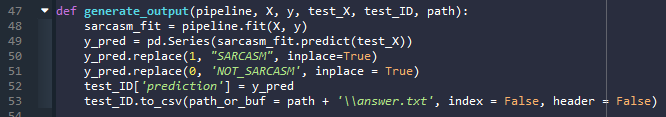
For a more efficient experiment, we built a pipeline to do this process. As shown in the code below, the pipeline is composed of two parts, a vectorizer and a classifier. The function also helped set key parameters of the vectorizer.



Once the pipeline constructed, we can feed training and evaluation data into the pipeline by calling function “accuracy\_summary” to check the accuracy report of the current model. We can also tune the model by recurrent calling this function.

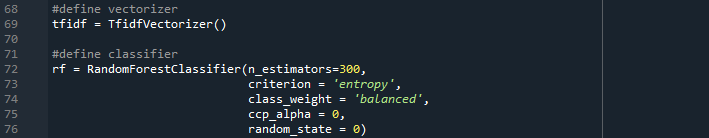


If the model is fine-tuned, we may call the function “generate\_output” to retrain the model with all training data and make the prediction on the testing data. The prediction result will be automatically generated as “answer.txt” which can be feed into LiveDataLab for further testing.



The main process is simple by following the three steps:

First, define tokenizer and classifier:



Second, build the pipeline with defined tokenizer and classifier:



Third, train the model and predict the result by feeding the data into the pipeline:



iii. Results

We chose TF-IDF as the vectorizer and we also chose three popular machine learning models as the classifier. The F-1 Score measure was used for evaluation. The results are shown in the figure below:

The results indicate all models overfitted data no matter how we added regularization terms which inferred the tokenized vocabulary matrix should be modified to ensure only key-performance words are counted. Since none of test results surpassed the baseline (0.723), we will no longer focus on traditional machine learning methods and move forward to neural network based models and BERT.

**BERT**

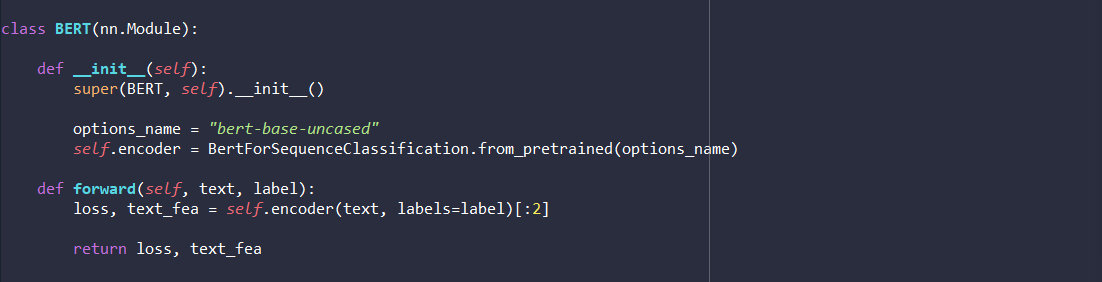
The best model that we have try is bidirectional encoder representations form transformers (BERT) model. BERT is a transformer-based model for NLP pre-training developed by Google in 2018. Unlike other models, BERT reads entire sequence of words in sequence from left-to-right or right-to-left at once. This bidirectional text input method takes the consideration of syntactic aspect of the language.

The BERT model that we used in this project is “bert-base-uncased” version. This model is a smaller model that consist with 12-layer, 768-hidden, 12-heads and 110M parameters). We decided to use this model is because this model is small enough that we can run in our own PC and the data size for our project is not very large. Since transformer can be used in PyTorch, so our iterators and models are running in GPU.

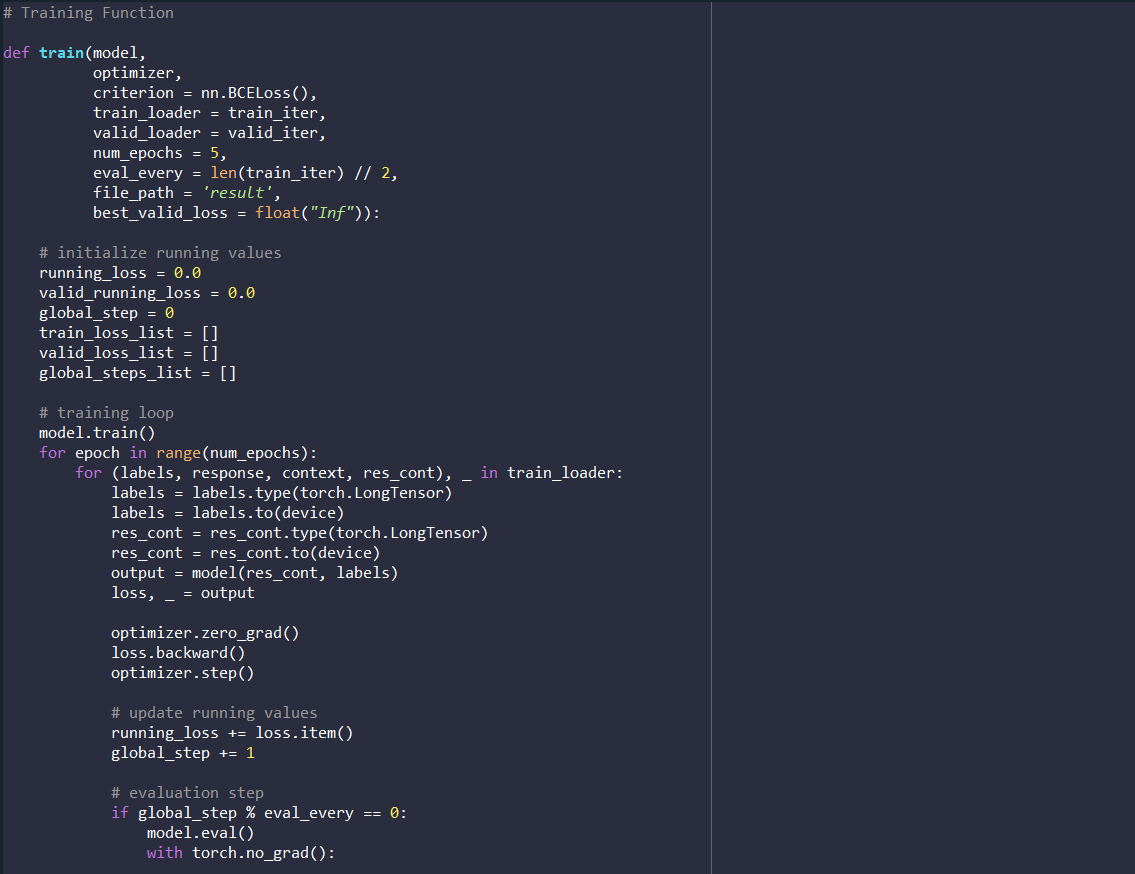
To implement this model, we need to prepare for the dataset first. Since we want to analyze both response and context to determine whether the tweet is sarcasm, we concatenated response and context column together as res\_cont column. After that, we divided training data into 2 parts randomly. First part has 4000 data for training. The rest 1000 data is for validation and test. Those datasets are then tokenized by using “bert-base-uncased” method and divide into each iterators, and each iterators has batch\_size of 16. See the code below.



As we mentioned before, the model that we used is “bert-base-uncased.” The code below shows the model build.



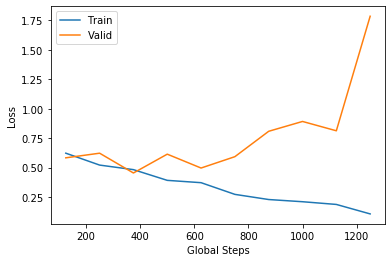
We then train the model by following code. The optimizer that we use as Adam and we tune BERT for 5 epochs.



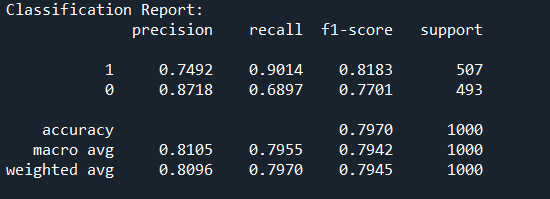


Since each model is very large (like 400MB+), we only save the optimizer that has best training loss, validation loss, and global steps.

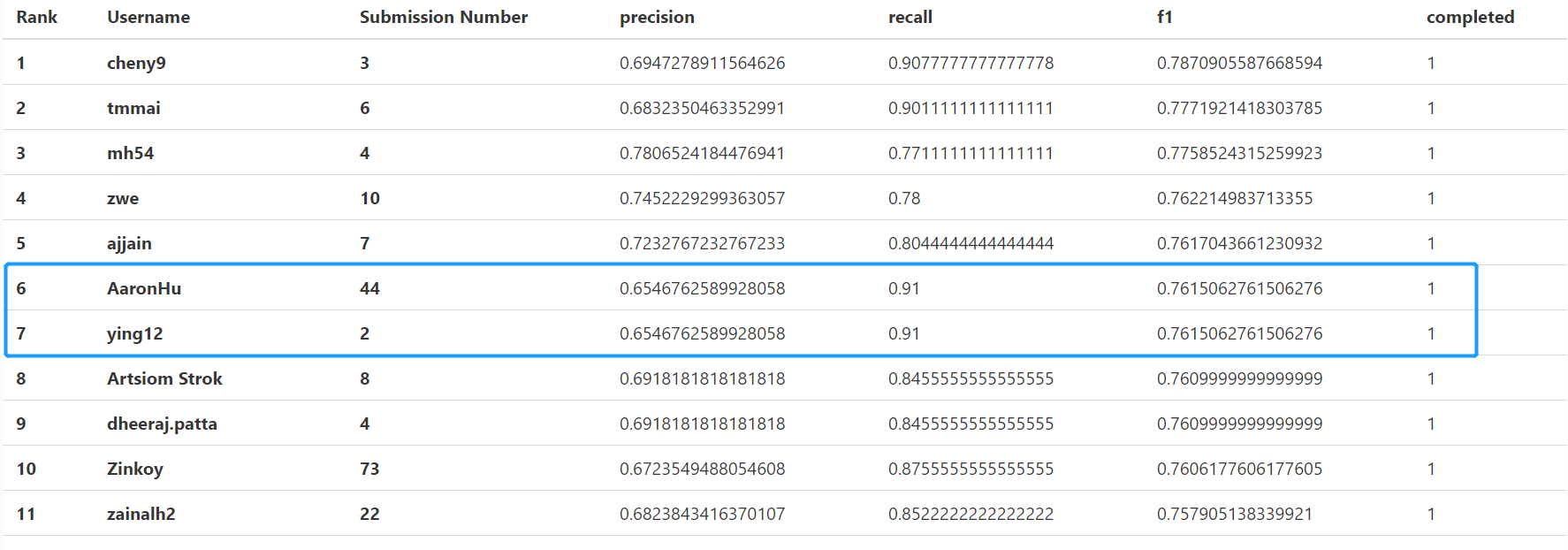
The line graph below shows how the loss of training and validation dataset changes at each global step.



After that, we use the test dataset that was separated from total training dataset to calculate the accuracy of the result. The table below are the result that we got from the model. As you can see, our overall accuracy is 0.7970, which passed the baseline of this competition.



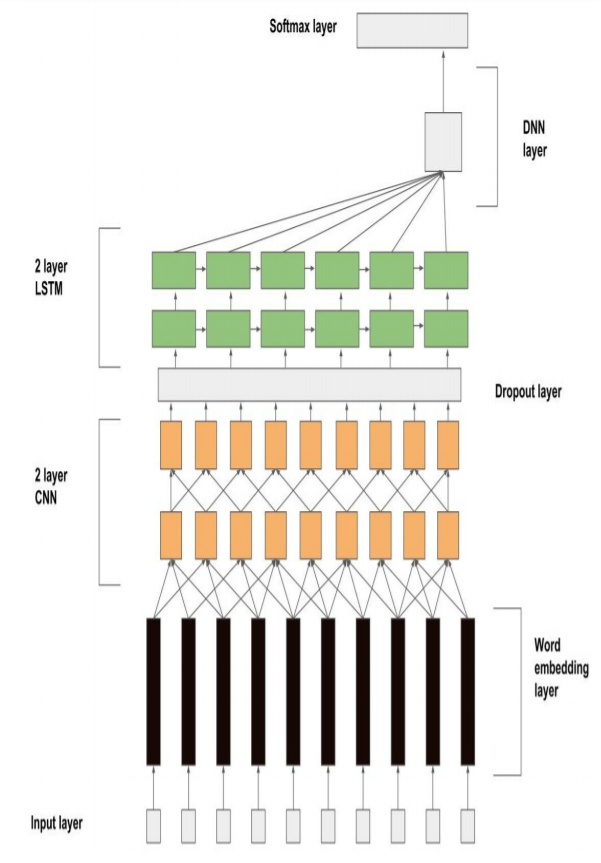
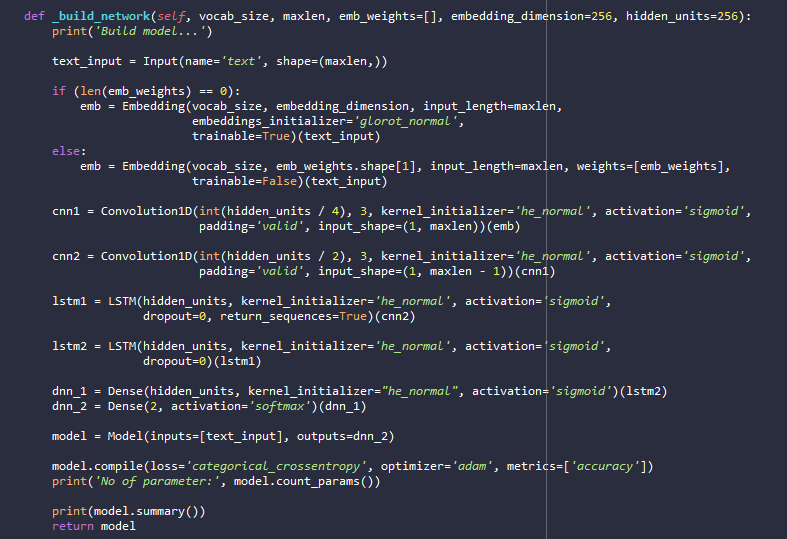
Lastly, we use this model to predict actual test dataset. We have passed baseline and placed top 10 of the leaderboard.



**CNNs + LSTM + DNNs**

With the recent move to artificial neural networks in NLP, ANNs provide an alternative basis for semantic modelling. Therefore, we tried a neural network model for semantic modelling in tweets that combines Deep Neural Networks (DNNs) with time-convolution and Long Short-Term Memory (LSTM).

In our experiment, apart from the combination of CNN, LSTM, and DNN, we observed the performance for each of the neural networks individually. The CNN network is investigated by varying the number of filters and the filter widths, set to 64, 128, 256 and 2, 3 respectively. For the LSTM network, the number of memory units is varied from 64 to 256. Sigmoid is chosen as activation function for both networks. We used Gaussian initialization scaled by the fan-in and the fan-out for the embed ding layer and Gaussian initialization scaled by the fan-in for the CNN, the LSTM, and the DNN layer as initial probability distribution. The code was implemented using keras library. The codes and figure below show the detail of this architecture:



To train the model, we simply feed tokenized data into the model. Our experiments indicated with a batch size of 64 and epochs of 7, the model would perform the best.

**Results and Conclusions**

Finally, we combined all results and made the comparison.

The results proved that BERT is one of the most powerful NLP model at present and can be easily modified and applied on different NLP tasks, which also indicates the pre-trained model played an important role in improving the accuracy. The neural networks are also powerful. However, the outputs of neural networks are unstable and hard to generalize. In other word, a well-trained neural network may not perform well on other datasets, despite the same task.

**Reference**

SarcasmDetection by AniSkywalker: <https://github.com/AniSkywalker/SarcasmDetection>

Aniruddha Ghosh, Tony Veale. Fracking Sarcasm using Neural Network.

https://arxiv.org/abs/1810.04805 Pre-training BERT

https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b Guide for BERT classification

NLP-with-Python by susanli2016: https://github.com/susanli2016/NLP-with-Python/blob/master/Latent%20Semantic%20Analysis\_reviews.ipynb

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, Prateek Vij. A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks

Akshay Khatri, Pranav P. Sarcasm Detection in Tweets with BERT and GloVe Embeddings.