Text Classification Competition

The main goal of this project is to develop a learning system that predicts the label of response on Tweet to be 'SARCASM' or 'NOT SARCASM' while optionally using the context by learning from responses labeled already. The trained model should achieve a fa score above 0.723 on the test data. We trained a machine learning model first, which is XGBoost as our baseline model. It can only achieve a 0.65 f1 score and overfit a lot. Therefore, we have been focused on deep learning models and tuning the parameter after more than 140 times, and we finally beat the baseline.

We used the Convolutional Neural Network (CNN) to accomplish this text classification task and achieve an accuracy of 0.7247, which beats the baseline of 0.723. CNN is a class of deep neural network that can take in an input, assign importance to various objects in the input, and be able to differentiate one from the other. It can automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a particular predictive modeling problem. We utilized Python and Jupyter Notebook to develop our learning system. The libraries related were Json, Pandas, String, Nltk, Numpy, Matplotlib, Sklearn, Re, Keras, GENISM, etc. The steps are as follows:

1. Load data: We loaded the data line by line and converted the raw train.jsonl data to a data frame called "reviews".
2. Clean and pre-process the data: We made a clean function to remove URL text such as http, @, #, and any numbers. The information from "response" and "context" was combined into a new column: "text". Then we tokenized the texts by using NLTK's word\_tokenize so that a sentence is divided into single words. We also turned all letters to lower case. Next, we added two new columns to the 'reviews' data frame to prepare for the binary classification.
3. Prepare Tain and test sets: There were 5000 objects to train the system and 1800 objects to test. We built training vocabulary and got maximum training sentence length (73) and the total number of words in the training set (12526).
4. Load Google News Word2Vec model and trained word embeddings: After that, we loaded the Google News Word2Vec model and trained our word embeddings.
5. Tokenize and Pad sequences: We assigned an integer to each word and put that integer in a list. We padded the sentences so that all training sentences had the same input shape (50). We got embeddings from the Google News Word2Vec model and saved them corresponding to the sequence integer assigned to each word. If there were no embeddings, a random vector was saved for that word.
6. Define CNN: the content of 'text' as a sequence was passed to a CNN. The embeddings matrix was passed to embedding\_layer. We applied five different filter sizes to each content and GlobalMaxPooling1D layers to each layer. Outputs were concatenated. Dropout layer - Dense layer - Dropout layer - final Dense layer was applied. We also printed a summary of all the layers with corresponding output shapes.
7. Train CNN: The number of epochs and batch size we utilized were 3 and 80, which means our model will loop around and learn three times, and eighty data will be viewed at a single time. Since the training dataset was small, we took this relatively small number of epochs to avoid overfitting. Since sarcasm is hard to detect, we also lower the confidence level from 0.5 to 0.38 during the inference phase, which means if the classifier is 38 percent sure this sentence is sarcastic, then this sentence will be predicted as sarcasm.
8. Test: We used the model to predict the label of data in the test set and get an accuracy of 0.7247.

The CNN system could be utilized on other tasks, such as image recognition, Electromyography (EMG) recognition, which relates to identifying human intention to control assistive devices, video analysis, drug discovery, health risk assessment, and biomarkers of aging discovery, etc.

Except for the CNN model, we have also tried another deep learning model: BERT (Bidirectional Encoder Representations from Transformers) model, though it didn't beat the baseline score. Bert model is a pre-trained model, and it typically has several advantages over other NLP models, including quicker training, smaller data size requirement, and higher prediction result. It is widely used in text feature extraction and text classification tasks. However, it performed well in our validation datasets but had inferior prediction results on the test data, which was surprising.

To run a BERT model, we utilized the free GPU offered by Google Colab. The libraries we mainly used were TensorFlow, Torch, and Transformers package from Hugging Face library. We used the same data cleaning approaches as the CNN model and did not remove the stop words. We then used the pretrained Bert tokenizer method to convert the response into tokens. Each response was split into tokens of the same length. The shorter response text would be padded with token zero, and we added attention masks to differentiate the padded tokens from the true tokens. The maximum response size was 85, and we set the fixed size to be 120 in case the size of test text data is larger. We split the 5000 training data into 80% training data and 20% test data. The classification model we chose is BertForSequenceClassification. We tried to tune the hyperparameters such as batch size, learning rate, and the number of epochs. However, the best-tuned parameters made the test prediction results worse. In the end, we used batch size = 16, learning rate lr = 2e-5, eps = 1e-8 and epochs = 2. The metric we used to evaluate was the f1 score. When validating the model results from training, the f1 score we had was around 80%. Finally, we did the same procedure to the test data before modeling, and as mentioned above, the results were surprisingly not satisfactory. Data cleaning techniques and tuning parameters also didn't improve the results. Future attempts might include trying different BERT models and padding choices.

**Reference:**

CNN: https://towardsdatascience.com/cnn-sentiment-analysis-1d16b7c5a0e7

BERT: https://colab.research.google.com/drive/1pTuQhug6Dhl9XalKB0zUGf4FIdYFlpcX#scrollTo=DEfSbAA4QHas