Text Classification Competition: Sarcasm Detection Final Report

Chua Yeow Long, ylchua2@illinois.edu

Introduction

In this competition, we are required to perform sarcasm detection/classification of twitter tweets. The dataset consists of a response which is the tweet to be classified, context which is the conversation context of the response and the label as well as the id for identification of tweets when making submissions. The goal is to predict the label of the tweet "response" while optionally using context (i.e., the immediate or the full context). From the size of the dataset which is 5,000 for training and 1,800 for testing, transfer learning models will need to be used instead of building neural network models from scratch.

Installation of libraries

I'll use jupyter notebooks to illustrate the workflow and we'll need NLTK for text pre-processing, Keras/TensorFlow to build our neural networks and standard sklearn libraries. To install the libraries, just do a pip install or conda install (in the case of anaconda).

```
pip install tensorflow-gpu
pip install nltk
pip install sklearn
```

Overall Plan

I'll first preprocess the text data by removing stop-words using NLTK's stop-words and non-alphabetic characters such as symbols before feeding the data into 3 different models. The first is to use a simple word occurrence/count model and the model did not work out very well as the model predicted not-sarcasm for all the test dataset. The second is to use GLoVe embedding and adding some LSTM and dropout layers and training just the newly added layers. A 85% train and 15% validation split was performed to obtain the validation dataset. For the input text data, I have tried with and without context and there was not much a difference in the results. The last and third method is to use BERT and re-train the entire BERT layers using the dataset on a pretty, I would say beefy machine. The best F1-score I have obtained is 0.7378 with context information and 0.716 without context which is pretty significant. I have explored using additional feature engineering and GPT-2/XLM models but given the time constraints, I was not able to obtain a better F1-score.

Code Walkthrough

For this section, you can just refer to the accompanied jupyter notebook or the video presentation.

Results

I have not included the results for the CountVectorizer model as it predicted all 0's/Not-Sarcasm. The best F1-score I have obtained using GLoVe embedding together with LSTM layers was 0.661 both with and without context. The best F1-score I have obtained using BERT is 0.7378 with context information and 0.716 without context. The baseline required is 0.723.

Description	F1-score
GLoVe embedding with LSTM layers without context	0.661
GLoVe embedding with LSTM layers	0.661
BERT without context	0.716
BERT with context	0.7378

Conclusion

It was a fun journey to be playing with the state-of-the-art in NLP, playing with BERT and GLoVe embeddings as well as GPT-2/XLM. Although I was able to surpass the baseline only using BERT, I believe with more feature engineering and tinkering with the models, the baseline should be achievable.