Project Documentation | Classification Competition

This is an individual project. Samuel Valenzuela (samuelv4@illinois.edu)

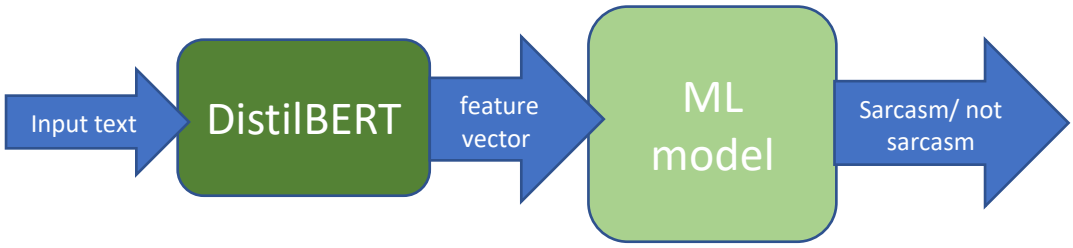
I followed two different approaches for the classification competition. One was using BERT as an embedding layer and the other was to train BERT and use it as a classifier. The second attempt was the one that performed better than the baseline and the one I document more in detail in the notebook with a clear structure. I also provide the notebook for the first approach but is not very well documented or easy to follow as I tried many different things and the code is a bit of a mess.

To run the models I recommend to use Google Colab. I always used Google Colab for this project as it runs much faster than on my laptop. It also avoids having to install libraries or dealing with python environments.

First approach: sam_disbert.ipynb
Second approach: sarcasm_transformers.ipynb

BERT as embedding layer (sam_disbert.ipynb)

My first idea to solve the classification problem was to use a pre-trained model as an embedding layer that turns the text into a vector of numbers that can be used as features for a Machine Learning model. I use BERT to get high quality features out of the data. Because BERT has been already pre-trained with massive general datasets, it is useful for using it as an embedding layer that takes the tweets and outputs a vector of numbers that can be used as features.



Here I list the different variations I tried:

- BERT base and DistilBERT. I tried both models for the embedding layer. DistilBERT worked better.
- Logistic Regression as ML model. It worked very well obtaining a 0.69 f1.
- Linear SVC as ML model. It obtained around 0.66 f1.
- Random Forest as ML model. It worked the best with a 0.712 f1. Very close to the baseline.
- Convolutional Neural Networks. I tried many different networks with different number of layers and neurons. It worked also well but didn't get a higher f1 than Random Forest.
- Added a subset of data from Ghosh dataset for training. This dataset has over 50,000 tweets. This didn't help at all and the f1 was lower. Maybe the type of sarcasm is different in some way in each dataset.

Fine tune transformers (sarcasm_transformers.ipynb)

My second attempt was about actually training pre-trained models and using it as a classifier instead of an embedding layer. Here I only use the transformer and not an additional ML model. I tried different transformers provided from the SimpleTransformers library including BERT, DistilBERT, XLM, XLNet and RoBERTa.

Most of them performed better than the baseline. The one that showed better and more consistent results is RoBERTa with 0.74 f1 score.