**Report**

The purpose of this project was to to design a state-of-the-art model to solve the problem of entity-based sentiment analysis.

I had previous experience with sentiment analysis (not entity-based) by using word embeddings and RNN with LSTM units. I was very excited to research and implement the state of the art in entity-based sentiment analysis by using Bidirectional Encoder Representations from Transformers (BERT). Of course, I had to do research and study the original paper published by researchers at Google AI Language about BERT and how to implement in a real project.

The first problem I encountered was how to include into the training the targeted entity so that the model can learn the relationship between the sentiment and the targeted entity. I found an interesting approach used by companies to understand sentiment from articles. The article was called ‘*News to Company Linking with Bert’* (<https://towardsdatascience.com/news-to-company-linking-with-bert-48a1ac9805f1>). In this article the authors used the idea of including the targeted entity in the beginning of the main text with the separator ‘|’ between entity and article text. This idea is arbitrary, and they expected the model to learn this key ‘|’-context semantics after seeing enough samples. They concluded that ‘*The results are pretty impressive for a Bert-only model without any tweaking or smart tricks’*. I decided to use the same trick and see how well the model will perform.

Next, contrastive conjunction and negation was handled by not removing stop words and thus letting the model to understand the meaning of contrastive conjunction and negation in the sentences together with the previous mentioned trick. In ‘Natural Language Processing in Action’ by Lane et. al (2019) the authors also suggested not to remove stop words during pre-processing since it reduces important information and connection between words. So, I pre-processed the text of the training and testing set the same way (normalizing it, remove non characters, etc.) and then used the technique previously mentioned to include the targeted named entity in the main text.

One of the limitations of BERT is the lack of ability to handle long text sequence (BERT support up to 512 tokens). In addition, BERT is categorized as autoencoder (AE) language model. It uses the [MASK] in the pretraining, but this kind of artificial symbols are absent from the real data at finetuning time, resulting in a pretrain-finetune discrepancy. Another disadvantage of [MASK] is that it assumes the predicted (masked) tokens are independent of other unmasked tokens. This is not true in real natural language and thus this is another limitation of the model.

Back to our task, I had never used BERT before, so I had to learn how to transform our text in order to use it as an input to the BERT network. Using the BERT tokenizer I tokenized the text and transformed the text into integers, using specific tokens to indicate the beginning and end of a phrase. I encountered a problem when using StratifiedKFold in the training of the network, so I manually split the training and validation sets. If I had more time, I am confident I would figure out what caused the problem and use StratifiedKFold. I had to take a decision and compromise in the limited amount of time I had.

**The ‘logically.py’ file contains extended commentary for every line explaining the purpose and the functionality of the code**. During training I calculated a lot of statistics such as: accuracy, loss, F1-score, precision and recall. During training and validation, I saw that by using 4 epochs I was clearly overfitting and epoch=2 was found to be the best.

The results in the validation set were very positive with accuracy=94%, F1-score=95%, Precision=96% and Recall=95%. It would appear that the strategy I followed to include the targeted entity in the main text was very effective indeed. Finally, I calculated the predictions for the test set ready to be used to calculate the test scores later.

For the project I used Pytorch and Google Collab in order to take advantage of a strong GPU.

I enjoyed this challenge immensely and I look forward to discussing it with you.