# Logically Targeted Sentiment Analysis

## Code

The code is available on GitHub in the repository <https://github.com/priestleyn/logically>. The repository contains the train/test data so I’ve made it private. If it’s okay to make it public then please let me know and I’ll open it up (contact me at [nicholas.priestley@gmail.com](mailto:nicholas.priestley@gmail.com)). Alternatively, provide GitHub usernames and I’ll add them as collaborators to the project.

The solution was developed using Jupyter notebooks on Google Colab Pro using a GPU Runtime. Instructions for running the code and obtaining the results are in the README.md. Results are also saved in the repository (‘Entity\_sentiment\_testV2\_Results.csv’).

## Approach

Due to lack of experience in this domain, some research was required to get me going. The following paper <https://arxiv.org/pdf/1905.03423.pdf> was helpful, it provides some background on Targeted Sentiment Analysis (TSA), disambiguation of terms, popular research datasets etc. - one of which I used when evaluating model (SemEval2014).

At the time of writing, most state-of-the art NLP research that I’m aware of makes use of transformers. Also, I have prior experience using BERT, it’s simple and there’s lots of content describing its usage so my natural inclination was to find a paper that uses BERT for TSA.

I came across ‘Utilizing BERT for Aspect-Based Sentiment Analysis via Constructing Auxiliary Sentence’ (Chi Sun et al., 2019) <https://arxiv.org/pdf/1903.09588.pdf>. It uses an incredibly simple (almost suspiciously so) approach, using BERT sentence pairs to frame the problem. In a nutshell, BERT is fine-tuned with the ‘Sentence’ in one side of the pair, and the ‘Entity’ in the other pair. Two main approaches are used to construct the features: pseudo-sentence (basically the raw data with separators), and question answering. See examples below:

* Pseudo sentence - [CLS] The food was good [SEP] food [SEP]
* Question answering - [CLS] The food was good [SEP] What did you think of the food of it? [SEP]

Not only are sentences separated with tokens, but also with a binary mask, specifically the ‘token\_type\_ids’.

A fully connected layer is added, and the model is fine-tuned. My approach largely follows this paper.

## Evaluation

My favoured metric for Binary classification is AUC. It uses rank ordering of probabilities to capture the degree to which the two classes are separated - though it doesn’t factor in model alignment so I’ve also recorded the accuracy (more on that later). Pretty much all approaches I tried converged in 2-3 epochs and performed very well on the validation set (20%). Approx. 0.99 AUC.

Though I didn’t do much quantitative analysis on the test set, I could see that the data wasn’t completely representative of the training data. E.g. 50 of the 255 entities were non-matched with the training data. Also, by eyeballing the data I could see that there were quite a lot of opposing sentiment Sentences. For that reason, I came up with a couple of independent datasets for evaluation of my model. SemEval2014 and a set of approx. 30 tricky observations from the test set that I hand labelled.

My final model choice was based on performance of these two datasets. Despite not having sight of the SemEval2014 training data, my final model nearly matches the performance stated in the Paper i.e. 0.94 AUC vs. 0.96.

## Questions

### How does your model handle contrastive conjunction and negation?

No specific techniques were used to handle contrastive conjunction and negation. The promise of BERT is that representations learned by the model capture the nuance of language through language modelling on the vast corpus i.e. BooksCorpus (800M words) (Zhu et al., 2015) and English Wikipedia (2,500M words) (see BERT paper <https://arxiv.org/pdf/1810.04805.pdf>). That said, with more time and resources it would be good to empirically test this hypothesis. I suspect that with the right training data and fine-tuning this could be improved.

### How do you deal with sentences that have multiple named entities and opposing sentiment?

I made some attempt to cater for this. I augmented the training data by concatenating opposing sentences for half of the training sample. E.g.

<The customer service was brilliant>. The website was slow | website | Negative

Where the text in the angle brackets was inserted from another opposing sentiment observation. This was done randomly with random order i.e. <pos> neg OR neg <pos> and vice-versa.

I did some very basic qualitative evaluation to measure the effect of this by reviewing the outputs of hand-written sentences that were given to the model (before and after modifying the training data). I also used the approx. 30 hand labelled observations from the test data (see ‘test\_small.csv’) as a guide - a large proportion of these observations have opposing sentiment. Though it is only a tiny sample, the augmented version of the training data performed marginally better on this dataset.

### How do you handle unseen vocabulary?

No specific techniques were used to handle unseen vocabulary. WordPiece embeddings (Wu et al., 2016) go some way to helping with unseen vocab, also the large token vocabulary of 30,000 makes it less likely that you’ll come across a new token. It’s possible to extend the tokens if required.

## Limitations

* The distribution of the probabilities is aligned with the oversampled training data that I used to create the data augmentation for opposing sentiment. Though this doesn’t hurt the rank ordering of the probabilities (i.e. AUC is still strong), some work may need to be done to choose an appropriate threshold for pos/neg. See probability distribution of the test set in the notebook – it’s artificially balanced. Accuracy threshold was set to 0.5 in my results.
* There’s no native support for k-fold cross validation in PyTorch. With more time I’d look at using something like SKorch which is a high-level API for Pytorch that provides the functionality.
* The model is not trained to handle cases where the ‘Entity’ isn’t contained within the ‘Sentence’.