Irony Detection

My approach to irony detection experimented with input features and model architectures, and I cascaded any improvements I made throughout following models.

I first hypothesised that a simple and effective method for irony detection would be to extract BERT sentence embeddings for each tweet, and to then train a feedforward network to classify those sentences as ironic or not ironic. After experimenting with hyper parameters, I found that a feedforward network with 3 hidden layers of size 50, a learning rate of 0.001, and batch size of 32 improved F1 to 3% higher than achieved by the benchmark. This is not a huge improvement, considering that the largest improvements in the challenge achieved around 12% higher F1 than the benchmark.

One reason BERT might not be performing well here is because it is not able to encode all of the input tokens. Twitter vocabulary is not necessarily like the vocabulary BERT is trained on, with variant spellings (e.g. 'soo' or 'soooo' instead of 'so') and words that are hashtagged (#not). If these words are not in the vocabulary, many will simply be inputted to the model as the unknown token, limiting the power of the sentence representation. Assuming this is the case, we can fix some of this by preprocessing. I used the Twitter Preprocessor used in the benchmark to preprocess the corpus before feeding it into Bert as a Service pipeline. Furthermore, before I preprocessed the corpus I removed all hashtags, so that some previously hashtagged words could be represented by a pre-trained embedding. These two changes combined to produce, on aggregate, just less than a 1% improvement, bringing the total improvement over the benchmark to around 4%.

Attempting to boost performance, I added some sentiment features using the VADER analysis tool. This was inspired by the team NLPRL-IITBHU as mentioned in the SemEval paper. One of the characteristics of ironic sentences is that they may have tokens corresponding to opposite sentiment polarities, where the writer expresses their unhappiness sarcastically (e.g. "Just love being anxious at 4 in the morning."). I added four sentiment values (overall sentiment, amount of neutral sentiment, amount of positive sentiment, amount of negative sentiment) to each BERT sentence embedding. This, however, produced no discernible improvement. Either the sentiment information included was already contained to some extent in the BERT representation or there was not enough data to properly utilise the information.

Finally, I hypothesised that even de-hashtagged, normalised Twitter data was not optimised for use in BERT. It was possible that a more simple model with more appropriate embeddings would improve upon my previous setups. For example, there exist publicly accessible word embeddings trained on Twitter data using GloVe. These embeddings might be better suited for the current dataset and produce a higher F1 score. For each sentence, therefore, I extracted word embeddings. I then trained an extremely simple unidirectional LSTM to take the GloVe word embeddings as input and produce, at the end of the sentence, a classification of 'ironic' or 'not ironic'. This resulted in a small improvement over my previous methodology of 0.2%. This is encouraging, given I performed no hyper parameter search or model tuning. It is likely further improvements are possible using this technique.

These experiments suggest two main avenues for future work. Twitter sentences are not automatically well represented by BERT. One way to ameliorate this is to more extensively preprocess the twitter sentences, such that a larger degree of tokens fall into BERTs vocabulary. This has the benefit of being able to harness BERTs exceptional power, but the shortcoming that the model will not be able to

capture any meaningful differences between variant spellings of words to help irony prediction. Alternatively, one could use a transformer that has been trained to encode sentences from Twitter with a Twitter-centric vocabulary (if such a transformer exists, or could be trained).

The alternative avenue is to steer away from sentence embeddings, and towards dealing directly with word embeddings. One possible reason that an LSTM using lexical embeddings outperforms a feedforward network using sentence embeddings is that former setup is better able to transmit explicit sarcastic cues that may be absent or vague in a sentence embedding. For example, one word that appears often in sarcastic Tweets is 'well', in contexts such as 'well that was fun.' It is possible that these sorts of lexical cues are transmitted well through LSTMs when the network itself is trained to transmit these signals. On the other hand, BERTs sentence representation may be somewhat generic, as it is meant to capture the general facets of the sentence, such as meaning, sentiment, likely context, etc. If it is more effective to deal with word embeddings than sentence embeddings, future work should focus on using lexical embeddings (whether BERT, GloVe, or otherwise) as inputs to the task of irony prediction.

Experiments	F1
Baseline	0.589
Sentence embeddings with tuned feedforward network	0.621
Refined sentence embeddings (dehashtagged, Twitter tokenised) with tuned feedforward network	0.629
Refined sentence embeddings + sentiments in tuned feedforward network	0.629
Lexical embeddings used in untuned LSTM framework	0.631168831168831

Table 1: Experiment results performed in the study alongside the baseline