Translation of Sarcastic Sentences for Improved Sentiment Analysis Accuracy

Katie Scholl KSCHOLL@CS.BROWN.EDU
Yanyan Ren YREN17@CS.BROWN.EDU
Michael Coppolino MCOPPOLI@CS.BROWN.EDU

1. Introduction

NLP is a fascinating subset of deep learning that is being used in countless applications and being researched in depth today. The fundamental concept of NLP is training a model to process and analyze large amounts of natural language data. Sentiment analysis in particular is a very interesting subset of NLP which tries to extract sentiment from a text input, usually by manipulating word embeddings in some capacity. However, there is often a large disparity between what we say as humans and what we actually mean; namely when using sarcasm. Can we improve sentiment analysis by training a model to detect sarcasm, and having it learn that the sentiment a sentence carries can sometimes directly contradict the meaning of the words in the sentence?

We will attempt to use a combination of work from existing papers and reinforcement learning to train a model to translate sarcastic sentences into sentences that contain the true sentiment of the sentence. For example, if we had the sentence "I love waiting for the bus for 2 hours!", it would be changed to "Waiting for the bus for 2 hours is not pleasant." With good results, this framework could be appended to existing NLP pipelines to preprocess corpora which contain some sarcastic content in order to increase accuracy of sentiment analysis, or even be built into messaging apps to ease understanding of sentiment for language-learners and those less familiar with the context-deprived environment of online communication.

2. Methodology

Our architecture operates under the assumption that sarcastic sentences have a literal positive sentiment and an implied negative sentiment. Thus our model aims to identify the words in a sarcastic sentence which cause the sentence to have a positive sentiment, remove them, and induce negative sentiment in order to synthesize a literal translation of the sarcastic sentence which will have a negative sentiment.

Our model contains three parts: sarcasm detector, sentiment classifier, and negative inducer. First, a corpus containing a mixture of sarcastic and non-sarcastic sentences is fed into the sarcasm detector, which obtains the sarcastic sentences for the next step. Second, the sarcastic sentences are fed into the sentiment classifier, which assigns each word a probability of having positive sentiment, filters out the strongly positive ones, and leaves the remaining neutral words part of the sentence. Lastly, the negative inducer takes in the neutralized sentences, and transforms the sentence from a neutral context to a negative context, attempting to fill in negative words for the presumably positive words removed.

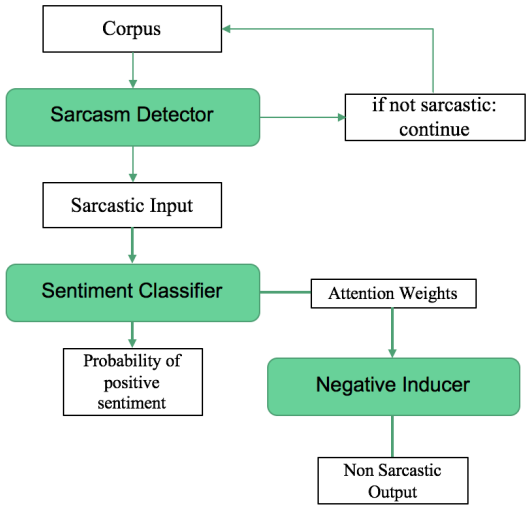

Figure 1. Model architecture

2.1. Sarcasm Detector

The sarcasm detector is used to determine whether or not a sentence in the corpus should be translated to a literal sentence, and for validation of the output of our model to determine if outputted sentences are sarcastic. It takes in a sentence and outputs a probability score, with scores closer to 1 indicating sarcasm and scores close to 0 indicating no sarcasm. The model uses an embedding layer with size 128, an LSTM layer of size 200 with attention, an LSTM

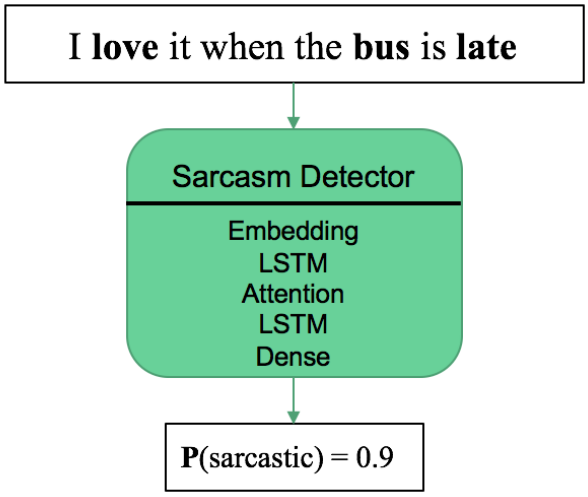

Figure 2. Sarcasm Detector

layer of size 150, and a dense layer of size 2.

2.2. Sentiment Neutralizer

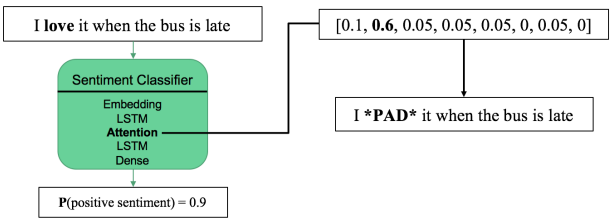

Figure 3. Sarcasm Detector

The sentiment neutralizer is trained as a sentiment classifier, which takes in a sentence and outputs a sentiment score, with scores close to 1 indicating positive sentient and scores close to 0 indicating negative sentiment. The classifier is trained using an embedding layer with size 128, an LSTM layer with size 200 with attention, an LSTM layer of size 150, and a dense layer of size 2 (CyberZHG).

2.3. Negative Inducer

The sentiment classifier is used as a neutralizer in the Negative Inducer module, which accesses the weights of the attention layer in the sentiment classifier calculated during the forward pass. For each word in the input sentence, the weights indicate the maximum attention paid to the word. We use the variance of the weights to filter out values that vary significantly from the mean attention in this vector. In order to access these attention weights, we used a slightly modified version of the SeqSelfAttention provided by keras-self attention (can).

The Negative Inducer follows a traditional seq2seq architecture and consists of an encoder and a decoder. The encoder and decoder both consist of an embedding layer followed by an LSTM layer, both with an embedding size of 500 and encoding size of 500. The outputs of the encoder LSTM and decoder LSTM are passed to an attention layer, and finally passed through a fully connected layer.

As the final fully connected layer required logits for each of the words in our vocabulary, we used the Tensorflow sampled softmax loss to avoid storing the entire tensor in memory through backpropagation.

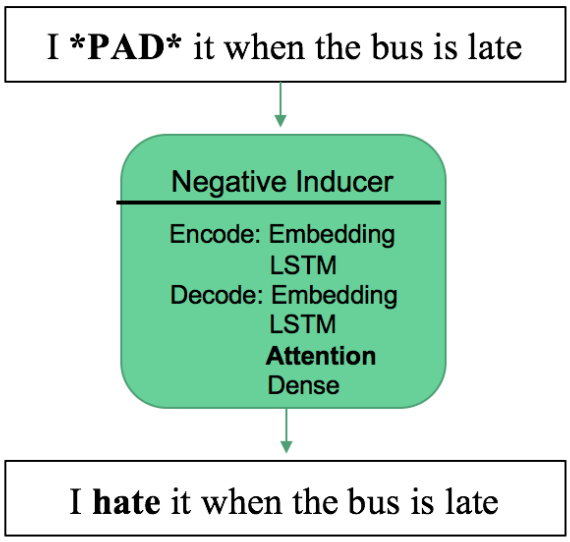

Figure 4. Sarcasm Detector

2.4. Datasets

We used three datasets: sarcastic (306,141 sentences), positive (47,827 sentences), and negative (47,827 sentences). The sarcastic sentences came from two sources. The first is the comments scraped Reddit comments containing the “” (sarcasm) tag. And the second source is the popular datasets used in many sarcasm detection papers, such as (Ghosh & Veale, 2016) and (Riloff et al., 2013). (2013), so there’s a high confidence that the sentences in the datasets are sarcastic. Positive and negative sentences are collected from four well-known sources that contains short snippets of strong sentiments: Stanford Treebank Dataset, IMDB reviews Dataset, Amazon Product Reviews, and Sentiment 140. The datasets have been preprocessed by Mishra et al.

(2019) to select out the positive and negative sentences that share the same vocabulary as the sarcastic sentences.

For the sarcasm detector, all sarcastic sentences are labeled as 1, and positive and negative sentences are labeled as 0. For the sentiment classifier, positive sentences are labeled as 1, and negative sentences are labeled as 0. The negative inducer takes in input of neutralized sentences produced by the sentiment neutralizer. It uses these neutralized negative sentences as input, and the original negative sentences as the expected output. For all models, the train-test split is 80%-20%.

We set the window size to be 30 and cut off long sentences. At first we did not do anything to restrict what goes in the vocabulary, and ended up with a giant vocabulary size of over 100,000 due to the usage of internet slang, spelling errors, and special characters. The large vocabulary size made everything train very slowly. Therefore, we removed all punctuation, replaced all non-English words with UNK tokens, and restricted the vocabulary to be the top 20,000 most frequent words.

3. Challenges

We spent a lot of time designing the architecture of the model. At the beginning, we planned to have three parts: a sarcasm decoder that translate sarcastic sentences into non-sarcastic ones, a sentence similarity checker and a sarcasm detector to evaluate how well the decoder worked – whether it changed the sentence to non-sarcastic and preserved its meaning. We were able to find existing papers on computing semantic similarity between sentences and pre-trained sentence encoder like Universal Sentence Encoder (ten), but none took sarcasm into account. To train a similarity checker that can recognize the same meaning between a sarcastic and non-sarcastic sentence, we would need a dataset with such pairs, but we had difficulties finding a large enough dataset. For the sarcasm decoder, we planned to train an embedding that contains sarcastic knowledge, and can substitute words based on distance in the embedding space. However, we realized that not knowing the structure of the high-dimensional embedding had led us to pursue a different path, especially since there is a high probability that sarcastic sentences are encoded in a very different way from their literal translations.

When we saw a paper on sarcasm generation (Mishra et al., 2019), we changed our plan. Their sentiment-based approach gave us inspiration to focus on the words with strong sentiment. Their model takes in non-sarcastic sentences and turns then into sarcastic ones, which is the exact opposite of our goal. After looked into the details in the paper, we realized that while the model was very modular and moderately well-documented, it was still a struggle to apply them to our topic, as there is a high overhead to interpreting the distributed, external-library-dependent source code.

4. Results

4.1. Sarcasm Detector

After training for 1 epoch and batch size of 100, the average accuracy for the test set is 91.52%. The training started out around 75% accuracy, which is fairly consistent with the ratio between the sarcastic and non-sarcastic labels (300 thousands and 90 thousands). The accuracy of the last batches in training is around 92%, which is similar to the final test accuracy, indicating that the model did not overfit.

4.2. Sentiment Neutralizer

The Sentiment Classifier accuracy is 96.18% after 1 epoch. The neutralization part of the pipeline, however, appears to be overly ambitious. While there are a few sarcastic sentences that were properly neutralized by our neutralizer, and provided enough information to infer the true meaning, most were stripped of enough context to make decoding back to the true meaning possible (see Figure 5).

4.3. Negative Inducer

The Negative Inducer perplexity after 3 epochs was 12 and accuracy 60%. We believe this to have been a fault in our design; looking through the neutralizer output, there is no way that it could have given an accuracy this high.

5. Reflection

Our architecture is well motivated and has an abundant potential for improving the feasibility of sentiment analysis on corpora with sarcastic content. There are several areas that our architecture could be improved, mostly involving the selection of data and approach to learning.

Sarcastic data is very difficult to gather, and it is hard to validate the authenticity of sarcastic data. Most of the sarcastic sentences we found were poorly written, with lots of grammatical mistakes and other issues that can not be fixed during preprocessing. Thus we ended up training our model with many misspellings replaced with UNK tokens, which reduce the ability for our detector to classify sarcastic sentences, and for our classifier to identify which words contribute to positive sentiment, as the words most likely to be misspelled are also the most likely to carry important semantic meaning critical to the semantics of the sentence. Our model's performance would be greatly improved with access to coherent and well collected sarcastic sentences.

An idea that could be implemented in the future is training the model with news headlines from satirical papers such as The Onion, and authentic sources such as NYT, WSJ, etc. Training the models with this data would solve the issue we had with our current dataset, as news headlines are guaranteed to be grammatically perfect and coherent, and we would receive perfectly labeled sarcastic data. However, it would be a challenge to compile a dataset of labeled positive/negative news headlines. In addition, news headlines are very context dependent, and are always written in the third person, which may cause poor performance when evaluating an input written in the first person.

A way that the learning of our models could be improved is via the implementation of reinforcement learning to update weights instead of labeled training. In theory the sarcasm detector could be used on the output of our model to calculate a sarcasm score, with the target average sarcasm of the output to be 0 and high sarcasm scores on the output indicating the models parameters have yet to converge.

"Failure is the condiment that gives success its flavor."
- Truman Capote

6. Link to Github

https://github.com/YanyanR/sarc2seq

References

Candidate sampling.

URL https://tfhub.dev/google/universal-sentence-encoder/1.

CyberZHG. Cyberzhg/keras-self-attention. URL https://github.com/CyberZHG/keras-self-attention/blob/master/keras_self_attention/seq_self_attention.py.

Ghosh, Aniruddha and Veale, Tony. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pp. 161–169, 2016.

Mishra, Abhijit, Tater, Tarun, and Sankaranarayanan, Karthik. A modular architecture for unsupervised sarcasm generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 6146–6155, 2019.

Riloff, Ellen, Qadir, Ashequl, Surve, Prafulla, De Silva, Lalindra, Gilbert, Nathan, and Huang, Ruihong. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 704–714, 2013.