Detection of sarcasm in text

Jørgen Nummedal Sveberg
Bacherlor in informatics / 2021
jorgenns@stud.ntnu.no

Abstract

Sarcasm is a huge part of our daily lives, and noticing the difference between a sarcastic news headline and a serious one can have large implications. Nowadays many people only scan headlines to affirm their believes or to stay on top of current events, or take things they read in a comment as fact. A lot of different types of data-sets will have different implications on how to approach this text analysis, and the rate of success vary widely. This paper will be exploring a few different approaches to detecting sarcasm in text, and comparing their viability. This will be in light of the data-sets used. The results show a comparison between simple and more complex neural networks, with and without context, and compared sarcastic news headlines to sarcastic reddit comments and tweets.

1 Introduction

Sarcasm in text is something even real people sometimes struggle with picking up on, but is a very vital part of sentiment and emotion analysis (Oprea and Magdy, 2019). Whether a comment, post, tweet or news headline is a joke or sarcastic can have large implications. We have more access to information and news than ever, and it is increasingly common to only read headlines and social media posts instead of the actual articles and research papers.

Because of this, the sarcastic nature of a social media post or a news headline is crucial. There is currently no way for the average person to know this differentiation at a glance, and artificial intelligence has huge potential in giving us tools to do the fact-checking for us preemptively.

A few different approaches to detecting this sarcasm has been attempted, such as Bag-of-Words/Bigrams, Sentence embedding, and LSTM neural networks. These all have different use-cases and varied success, and I will be discussing these during this paper. I will look at the difference in complexity and success regarding different the types of data-sets used for sarcasm detection, and the current methods used.

2 Background

The differences in datasets for sarcasm detection has large implications, as some types of text are more contextually dependent than others.

Depending on the nature of the dataset there is a varying level of accuracy and complexity. In Khodak et al. (2017) we can see that the CORPUS with 1.3 million entries achieve around 70-75 percent accuracy with a few different types of text analysis. These entries were scraped from reddit comments, where immediate context is very important, and it is therefore understandable that high accuracy is hard to reach. On the other hand, as seen in Mandal and Mahto (2019), sarcasm detection in more independent statements like news headlines seem to fare better. While more general context like political climate and current world affairs has an impact, simply analyzing the wording seems to have good results. I have also attempted some approaches to a data-set of tweets that utilized reactive supervision for aquisition of the entries, and much like the reddit comments, accuracy rarely went above 70 percent.

Utilizing these three different types of data, I approached the problem with a few different methods that i will explain now.

2.1 Vectorizer and Text embedding

To analyze text the neural network needs some way to express this in numbers. To do this we can use a few different methods, and I decided to attempt both with a `CountVectorizer` from the `sklearn` library for the simple method, as well as a `TextVectorizer` from the `tensorflow` library.

The `CountVectorizer` and `TextVectorizer` works by converting the words into a matrix of word counts. The `TextVectorizer` has a few more options, but both are mostly the same. The CountVectorizer will be used in the simplistic naive bayes solution while the TextVectorizer will be a layer in the tensorflow model.

2.2 LSTM

LongShortTermMemory is a recurrent neural network(RNN), and is useful for analyzing isolated context like for speech or handwriting recognition (Wikipedia contributors, 2021).

2.3 Naive Bayes BernoulliNB

BernoulliNB is probabilistic classifier, based on Bayes' theorem. This is suitable for discrete data, and is specifically designed for binary data.

3 Related Work

There has been some research into this field already, and is what I have based my own experiments off of. Khodak et al. (2017) Attempted a few different approaches including Bag-of-Words, Bag-of-Bigrams, and Sentence Embedding, which reached 70-75 percent accuracy on detecting sarcasm in their own Self-Annotated Reddit Corpus, which was based on reddit comments using the "s" tag. Mandal and Mahto (2019) used LSTM to detect sarcastic news headlines and approach 85 percent accuracy.

4 Architecture

A Few different architecture and data-set combinations were attempted in this project.

- Embedding and Bidirectional LSTM - With context
- Embedding and Bidirectional LSTM - Without context
- BernoulliNB - Without context

For the tensorflow NN, i decided to go with textembedding/vectorization, and bidirectional LSTM, with some dropout to fix overfitting. 1 shows the arcitecture used with context. The inputs here is the parent comment, and the sarcastic/non-sarcastic comment. The version without context is the same but with only one input.

The BernoulliNB version uses a simple sklearn BernoulliNB class with the CountVectorizer for pre-processing.

5 Experiments and Results

There were varying results depending on the architecture used, and the models that ignored context seemed to outpreform the ones who did by a large margin. The context based models seemed to overfit to an extreme degree, often reaching almost perfect training accuracy but only reaching close to 65 percent testing accuracy.

5.1 Experimental Setup

I have included the files used for a few different attempts, which can be used to run the same experiments.

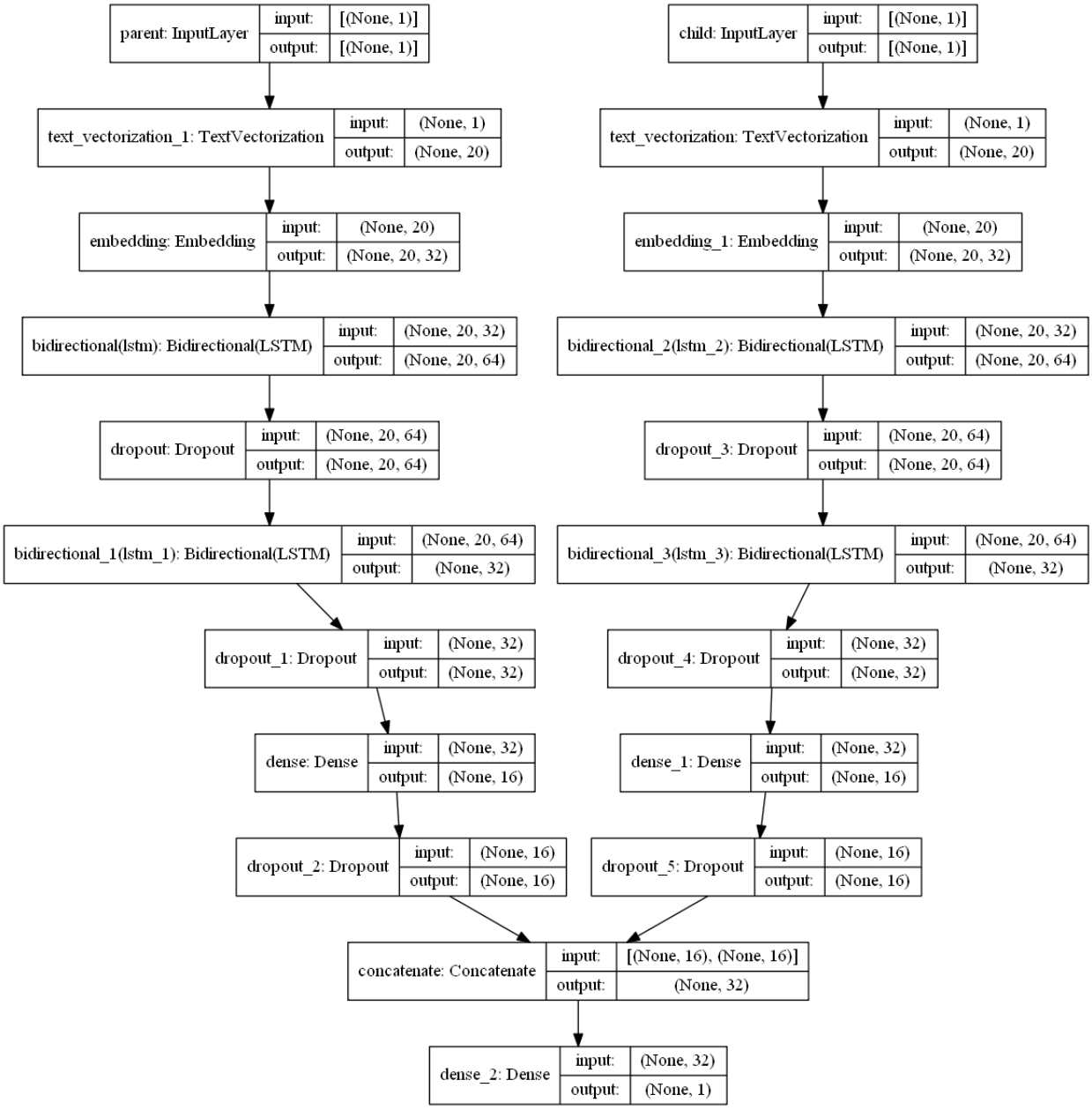

Figure 1: Embedding and Bidirectional LSTM

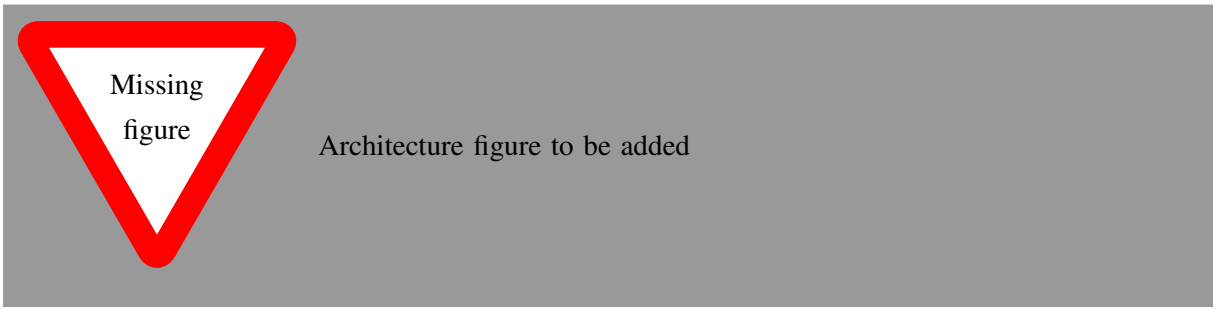

Figure 2: The missing architecture

5.2 Experimental Results

Results:

- No context embedding: Train accuracy: 0.9687 Test accuracy: 0.8581
- Context embedding: Train accuracy: 0.9971250295639038 Test accuracy: 0.6265000104904175
- BernoulliNB: Test accuracy: 0.77

The Bernoulli test results don't give us a training accuracy to compare, but it is obvious that a proper neural network utilizing LSTM has a better result, even if the training accuracy implies over-fitting.

6 Evaluation and Discussion

It is important to include a discussion, which describes what you have learned so far, the merits of the work as well as its limitations. It can be a separate section or it cab appear together with the results or be part of the conclusion). When evaluating your results, avoid drawing grand conclusions, beyond that which your results can in fact support. Further, although you may have designed your experiments to answer certain questions, the results may raise other questions in the eyes of the reader. It is important that you study the graphs/tables to look for unusual features/entries, and discuss these as well as the main findings. In particular, carry out an error analysis: What went wrong and why?

7 Conclusion and Future Work

What are the main contributions? How significant are they? Discuss the contributions in terms of the initial goal formulated in the Introduction.

Also consider how you think the work could be extended or improved, or what you could have done differently.

References

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. A large self-annotated corpus for sarcasm. *CoRR*, abs/1704.05579, 2017. URL `http://arxiv.org/abs/1704.05579`.

Paul Mandal and Rakeshkumar Mahto. *Deep CNN-LSTM with Word Embeddings for News Headline Sarcasm Detection*, pages 495–498. 05 2019. ISBN 978-3-030-14069-4. doi: 10.1007/978-3-030-14070-0_69.

Silviu Oprea and Walid Magdy. Exploring author context for detecting intended vs perceived sarcasm. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2854–2859, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1275. URL `https://www.aclweb.org/anthology/P19-1275`.

Wikipedia contributors. Long short-term memory — Wikipedia, the free encyclopedia, 2021. URL `https://en.wikipedia.org/wiki/Long_short-term_memory`. [Online; accessed 16-May-2021].