Sarcasm Detection in Twitter Data

Anan Aramthanapon
Harvey Mudd College
aaramthanpon@hmc.edu

Tatsuki Kuze
Harvey Mudd College
tkuze@hmc.edu

Saatvik Sejpal
Harvey Mudd College
ssejpal@hmc.edu

Abstract

The detection of figurative language devices such as sarcasm and irony is an increasingly important problem. We tackle this problem using the Twitter Dataset from the FigLang2020 Shared Task. Most previous tasks in this field tackle Sarcasm Detection using Deep Learning models like BERT, and also use context to improve results. This paper does not only that but also implements n-gram models both with and without context and discusses the differences in performances. Our results show that using BERT with Context gives us extremely promising results; however bigrams also do surprisingly well for a non-Deep Learning based model.

1 Introduction

Sarcasm refers to a use of words to convey meaning opposite of what is mentioned by the speaker in a literal level. As such, being able to recognize the use of sarcasm is important in correctly identifying the speaker's actual meanings, especially in context of applications like sentiment analysis. The classification task of sarcasm detection is considered a binary classification task, wherein the classifiers are provided as input utterances and possibly other contextual information to classify whether the utterance is sarcastic or non-sarcastic.

In particular, this paper will explore the question of how providing additional contextual conversational information to a sarcasm classification system affects the quality of the classification. The paper is driven by an existing shared task from the 2nd Workshop on Figurative Language Processing (FigLang 2020) at ACL 2020, and will use the Twitter Training Dataset provided in the shared task. (Ghosh et al., 2020) Nine models will be explored in this paper, three of which will consist of the BERT model transformer with a sequence classification linear layer on top (BertForSequenceClassification) (Devlin et al., 2019), and six of which will be based using a Naive Bayes classifier on n-grams features.

Of the three BERT based models, one model will be trained purely on the potentially sarcastic comment that is being classified, one will be trained on both the potentially sarcastic text and the preceding comments before that text. The final BERT model will take an alternative approach of sentiment analysis on both the response and conversational context utterances and using the resulting sentiments to detect sarcasm.

Of the six traditional n-grams based models, we have three models for unigrams, bigrams, and trigrams respectively trained on just the n-grams from the potentially sarcastic comment with tf-idf (Salton and McGill, 1986) weight adjustments and classified with a Naive Bayes classifier. We have three models trained on the same features with tf-idf and the same classifier, but also including n-grams from the preceding comments before the potentially sarcastic comment.

The nine models will be evaluated upon accuracy and F1 score, and further discussion will be made as to why each model may have performed better or worse compared to others.

The dataset we intend to use for this task is a set of Twitter Data from FigLang 2020's Shared Task. This dataset has the context for the tweet (the preceding comment), the potentially sarcastic response, and a label denoting whether or not a Tweet is sarcastic.

2 Literature Review

2.1 Related Work

One initial approach for such a task was done through a semi-supervised classification algorithm that used pattern extraction, selection and match-

ing and punctuation-based features for a kNN-like strategy. (Davidov et al., 2010) The paper analyzed Amazon reviews and twitter data that used a gold standard through Mechanical Turk, where each utterance was fed into the classification algorithm for a classification task. Another approach was taken in a research conducted on irony and sentiment analysis focusing on a corpus of ironic similes, acting as sarcasm markers. (Veale and Hao, 2010) It uses these markers to classify whether a simile is ironic or not depending on the lexical similarity between the terms and other factors like the frequency of its occurrence on the web. There were also other approaches that utilized word embeddings for the classification task, using distributional semantics methods and SVM classifiers. (Ghosh et al., 2015) In this method, the authors target a particular word to determine whether the sense associated with the word matches the remaining words in the utterance. While these methods have been successful in producing results that are beyond baselines, some more recent authors have researched alternative approaches.

One motivation for an alternative approach may be the problem that sarcasm detection is difficult. A paper was produced that noted the difficulty in sarcasm detection without context, which further implies that computers probably would also require more contextual information for a good sarcasm detection algorithm. (Wallace et al., 2014) The paper found that people when annotating comments for sarcasm requests more contextual information than for non-sarcastic comments. As such, it would make sense that contextual information would be a valid tool for analysis in classifying sarcasm through algorithms.

Context, which refers to any further information than the utterance in itself, (Joshi et al., 2017) was found to be used more and more frequently in recent papers (Ghosh et al., 2020). One approach utilized the user's historical tweets to predict sarcasm, where the user's past expression regarding the entities in the tweet of interest is utilized in the classification process of the sarcasm in a tweet. (Khattri et al., 2015) A majority voting based sentiment analysis regarding a target phrase within the tweet is used to determine the user's sentiment towards the target, and sarcasm is detected by observing whether the historical sentiment differs from the tweet's current sentiment. Another approach utilizes conversational context within Tweets in addition to the user's historical sentiments, audience information and other contextual information. (Bamman and Smith, 2015) Interestingly, the additional information regarding the audience did not have as much of an effect on the accuracy of the detection in comparison to the increase in accuracy from authorial information. A different paper uses a CNN to perform user profiling to provide the model with more context on the corpora, which was found to be helpful in this task(Hazarika et al., 2018).

An experiment similar to what we are proposing to do using a BERT encoder has been done in the past and has been shown to be highly effective (Van Hee et al., 2018). Using sentiment analysis for sarcasm detection has also been investigated by previous researchers (Abu Farha et al., 2021).

We will further investigate how traditional models such as Bag of Words or n-grams compare compare to using BERT.

2.2 Methods

The paper is driven by an existing shared task from the 2nd Workshop on Figurative Language Processing (FigLang 2020) at ACL 2020, and will use the Twitter Training Dataset provided in the shared task. (Ghosh et al., 2020) The Twitter Training Dataset consists of a series of 5000 tweets that were machine-tagged to be sarcastic if they contained the hashtags "#sarcasm" or "#sarcastic". Each tweet in the dataset is separated into "response" utterances and "context" utterances. The "response" utterances correspond to the actual utterances to be classified: the potentially sarcastic comment, while the "context" utterances are previous twitter posts to which the "response" utterance responded to.

For the non-sarcastic utterances, sentiment tweets are used proposed in related work (Ghosh et al., 2020), with sentiments with hashtags like #happy, #love, #sad, #hate, etc. The authors mentions that sentiment utterances were used since classifying sarcastic utterances against sentiment utterances is a harder task due to the nature of how many sarcastic utterances also contain sentiment terms.

For the data to be used for the model, we first tokenized the corpora using the BertTokenizer in the pytorch-pretrained-bert library. The BertTokenizer is used to tokenize the texts in the data into index numbers in the BERT vocabulary to be used for both the BERT and n-gram models. For models with context, we first joined the context utterances

to the response utterances before tokenizing. With the preprocess procedure finished, the data was first split upon training and validation data to be used for validation used in training.

BERT-based models in this paper are using BertForSequenceClassification (Devlin et al., 2019), after the data is converted into tokens using the tokenizer, the array of indexes were padded or truncated to match the maximum sequence length, which for this experiment was determined to be 128. Attention masks are then created for each data to also be used for BERT. The data then is converted into torch tensors, the required data type for our model. The torch tensors were then further converted into iterators with torch DataLoaders. The optimizer used for the machine learning training was BertAdam (Devlin et al., 2019), the optimizer frequently used in BERT models. For each model, the model was trained with 4 epochs in balancing time taken with results.

For n-gram based models, after the data is converted into tokens using the tokenizer, n-grams are constructed from the array of tokens and the counts of each n-gram is calculated for each instance, then weight balanced using tf-idf. We then train on this data using a Gaussian Naive Bayes classifier from the sklearn library.

For each of the three BERT models, the first model is the model that only is trained upon the "response" utterances. The second model uses the same approach, but with the merged "response" and "context" utterances.

The final model utilizes a different approach from the two previously mentioned models. It first utilizes a dataset of IMDB movie reviews (Maas et al., 2011) containing annotated data for sentiment analysis. The dataset is comprised from 25k training set data that contains annotations on whether a movie review is positive or negative. Using this data, a sentiment analysis model is constructed using the BertForSequenceClassification model with the pipeline mentioned previously. This model is then to be run on the twitter dataset, where the "response" utterance and "context" utterance are run sentiment analysis classification separately. The two sentiments are then later compared, wherein non-matching sentiments may suggest sarcastic within the "response" utterance. This approach is inspired by past works that discussed the connection between sarcasm and sentiment analysis. (Veale and Hao, 2010)

For each of the six n-gram based models, we will have 2 models each for unigrams, bigrams, and trigrams. One with just the "response utterances", and one with the "context" and "response" utterances merged.

We hope to gain insights about two main aspects:

- How does using context affect the accuracy of our model?
- How does using n-gram based models compare with using deep learning BERT based models?
- What is the optimal number of adjacent words to use for n-gram based models?
- Is sentiment analysis alone effective for detecting sarcasm?

By having models that use context, and similar models that do not use context, we are able to determine if having context impacts the accuracy of our models. These sets of models will provide us with enough information to allow us to analyze the impact of different lengths of n-grams, impact of context, impact of sentiment analysis, and the difference between traditional n-gram models and BERT models for the task of sarcasm detection.

Our first baseline experiment will be to use a model based on Bag Of Words along with tf-idf (Salton and McGill, 1986) using the Naive Bayes classifier, which is a baseline also used in similar experiments in the past (Hazarika et al., 2018). We will also experiment with a model based on n-grams and tf-idf. We will compare how these baseline models compare to a state-of-the-art deep learning BERT model (Devlin et al., 2019) for the same task. For each model, we will conduct two experiments: one with the preceding comment and the response, and one with just the response to see the impact context has on sarcasm detection.

Each of the nine models are then evaluated through testing on a validation dataset of 1000 datapoints. Each outcome of the classification algorithm will be compared with the gold labels, to compute F1 score and accuracy for each model.

To help visualize our results, we will have one figure that displays the words that show up most frequently in tweets that are categorized as sarcastic by our model, excluding stop words. We will have another figure that summarizes the differences in our models' performances with and without context, as well as using our baseline n-gram model

against BERT. We will also add a table to discuss certain testing examples for which our models had the worst performance, and try to analyze why our model didn't do well on those specific cases. We will also compare these results between models to see if there are certain features in some types of sarcasm that make them hard to detect in general or if there are certain areas in which a certain model is not effective at predicting.

3 Results

The performance of each of the predictor models were evaluated on a test set of 1000 data points, which were randomly separated from the training set in the beginning of the experiment. The test set was relatively balanced with 495 sarcastic and 505 non-sarcastic data points.

Model	F1	Precision	Recall	Acc.
Response Only	0.79	0.77	0.82	0.79
Context + Response	**0.94**	**0.96**	**0.92**	**0.94**
Sentiment Analysis	0.59	0.65	0.55	0.63

Table 1: Metrics for BERT Models

Model	F1	Precision	Recall	Acc.
Unigrams	0.54	0.62	0.47	0.60
Bigrams	**0.72**	**0.71**	**0.73**	**0.72**
Trigrams	0.70	0.70	0.71	0.70

Table 2: Metrics for BoW and Collocational Features with Context

Model	F1	Precision	Recall	Acc.
Unigrams	0.67	0.69	0.66	0.68
Bigrams	**0.70**	**0.72**	0.68	**0.71**
Trigrams	0.69	0.65	**0.74**	0.67

Table 3: Metrics for BoW and Collocational Features without Context

As shown on Table 1, of the three BERT models, Context + Response model did the best with a high value for each of the 4 evaluation metrics, with response following it and the sentiment analysis doing relatively poorly compared to the other two. Broadly speaking, it makes sense that the two BERT models that were trained specifically on the sarcasm task did well, since they can be thought to properly capture the language structures of the model through the pre-trained BERT model, and was further trained using the BERT sequence transformer for the specific task. The context not only is more information to be used for the classification, but also matches the results of (Wallace et al., 2014) wherein they showed how context is crucial information even for humans in sarcasm detection. As for the sentiment analysis, the relatively poor performance could potentially be from either that sentiment analysis isn't a great predictor tool for sarcasm, or that the IMDB dataset on which the setiment analysis predictor was trained on did not match well with the twitter data that was used for the task. To explore further, more specific analysis on examples where the predictors disagreed in their prediction will be analyzed.

Looking at the Bag of Words/Collocational Features (Bigrams, Trigrams) models that we used, we see from Table 2 and Table 3 that Bigrams trained with Context did the best out of all the n-gram models.

Additionally, while sentiment analysis alone may not be enough to effectively classify texts as sarcastic or not, it may be effective when combined with other features. Currently, we are tagging responses as sarcastic if the sentiment of the response does not match the sentiment of the context and not taking into account other factors. In the future, we can try to incorporate the features of sentiment analysis with other models such as our main BERT-based model and even n-grams based ones to see if this would improve its performance.

We found that adding context did not significantly improve any of the models and made the unigram model significantly worse, dropping the F1 score from 0.67 to 0.54. This makes sense as when combining the context and the response, our n-gram based models did not distinguish between n-grams that originated from the context and the response. As the context is usually not sarcastic, adding n-grams from this to a potentially sarcastic set of n-grams could confuse the model and lead to worse results. This is especially true for unigrams which analyze single words, and would require certain words to be classified as sarcastic words or non sarcastic ones. This is not as significant for bigrams and trigrams, likely because these models also account for word order more than unigrams and thus would be able to better understand the role

of the context n-grams. To make this distinction even clearer, we could also tag n-grams originating from the context and the response differently so that our models are able to tell them apart in the future.

Another important point to mention is that while making our n-gram models we did not remove stop words or punctuation while tokenizing the corpus. Removing stop words could potentially improve our n-gram results; however, as we did use tf-idf in our models it may not make a large difference.

Furthermore, we should also discuss the possibility that the gold labels for our dataset may not be entirely accurate. As was mentioned in the Shared Task, these tags were machine labeled based on whether the tweets contained the hashtags "#sarcasm" or "#sarcastic". Tagging sarcasm this way is not always accurate, and has actually been shown to be a poor way of tagging for sarcasm (Oprea and Magdy, 2019b). Additionally, this paper does not take into account the difference between perceived and intended sarcasm, which is an important distinction to make in sarcasm detection (Oprea and Magdy, 2019a), as something can be perceived as sarcastic but not intended to be, or vice versa.

References

Ibrahim Abu Farha, Wajdi Zaghouani, and Walid Magdy. 2021. Overview of the wanlp 2021 shared task on sarcasm and sentiment detection in arabic. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*, pages 296–305. Association for Computational Linguistics (ACL). The Sixth Arabic Natural Language Processing Workshop, WANLP 2021 ; Conference date: 19-04-2021 Through 19-04-2021.

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth international AAAI conference on web and social media*.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, page 107–116, USA. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Debanjan Ghosh, Weiwei Guo, and Smaranda Muresan. 2015. Sarcastic or not: Word embeddings to predict the literal or sarcastic meaning of words. In *EMNLP*.

Debanjan Ghosh, Avijit Vajpayee, and Smaranda Muresan. 2020. A report on the 2020 sarcasm detection shared task. *CoRR*, abs/2005.05814.

Devamanyu Hazarika, Soujanya Poria, Sruthi Gorantla, Erik Cambria, Roger Zimmermann, and Rada Mihalcea. 2018. CASCADE: contextual sarcasm detection in online discussion forums. *CoRR*, abs/1805.06413.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J. Carman. 2017. Automatic sarcasm detection: A survey. *ACM Comput. Surv.*, 50(5).

Anupam Khattri, Aditya Joshi, Pushpak Bhattacharyya, and Mark Carman. 2015. Your sentiment precedes you: Using an author's historical tweets to predict sarcasm. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 25–30, Lisboa, Portugal. Association for Computational Linguistics.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Silviu Oprea and Walid Magdy. 2019a. Exploring author context for detecting intended vs perceived sarcasm. *CoRR*, abs/1910.11932.

Silviu Oprea and Walid Magdy. 2019b. isarcasm: A dataset of intended sarcasm. *CoRR*, abs/1911.03123.

G. Salton and M. J. McGill. 1986. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. SemEval-2018 task 3: Irony detection in English tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 39–50, New Orleans, Louisiana. Association for Computational Linguistics.

Tony Veale and Yanfen Hao. 2010. Detecting ironic intent in creative comparisons. In *In: Proceedings of 19th European conference on artificial intelligence—ECAI 2010*, pages 765–770. IOS Press.

Byron C. Wallace, Do Kook Choe, Laura Kertz, and Eugene Charniak. 2014. Humans require context to infer ironic intent (so computers probably do, too). In *ACL*.