Detecting Sarcasm with RoBERTa

Cees Roele
cees.roele@gmail.com

Abstract

Sarcasm is a form of figurative language. To understand a sarcastic remark we should attribute a meaning to it that is other than the meaning of its words according to a dictionary. The present short paper addresses the notion of sarcasm, discusses a large dataset of sarcastic and non-sarcastic remarks, explains what kind of machine learning modelling would fit detecting sarcasm, reports on the results of an implementation of such modelling, places it in context with comparable efforts, and points out potential improvements of the offered system.

1 Introduction

Understanding how sarcasm works for people helps us to understand *why* an approach to detect it might be successful or not.

> *Sarcasm is an ironic or satirical remark tempered by humor. Mainly, people use it to* ***say the opposite of what's true*** *to make someone look or feel foolish. For example, let's say you see someone struggling to open a door and you ask them, "Do you want help?" If they reply by saying, "No thanks. I'm really enjoying the challenge," you'll know they're being sarcastic. Sarcasm is* ***all about the context and tone of voice****, which is why it works better verbally. It's something you'll know when you hear it.*[1]

This sounds a bit alarming to the prospect of successfully detecting sarcasm on the basis of a textual dataset: *seeing* someone or *hearing* their tone of voice are not available. We need particles of text to identify the occurrence of sarcasm.

[1]Emphasis added. https://examples.yourdictionary.com/examples-of-sarcasm.html

Let's read on a bit in the source of the previous quotation:

> *Sarcasm can come in all different types. Some are easier to catch on to than others.*
>
> - ***self-deprecating*** *- where you poke fun at yourself*
> - ***deadpan*** *- sarcasm given in serious tone*
> - ***brooding*** *- saying the opposite of what you mean in an irritated tone*
> - ***juvenile*** *- obnoxious statements that might come across as annoying*

Whereas *tone of voice* seems to refer exclusively to a sound, *serious tone* and *irritated tone* might involve specific wording. Could *poke fun at yourself* and *obnoxious statements* also be about used words?

In (Das and Kolya, 2021) we find that the following words have a high occurrence in sarcastic comments: 'oh', 'know', 'like', 'yeah', 'well', 'go', 'right', 'think', and 'really'.

Yet, we see little here of the *self-deprecating*, *deadpan*, *brooding*, or the *juvenile*.

To make sense of all this, we should understand what people do and aim for in a conversation. When people use sarcasm to "say the opposite of what is true", they don't just want their listeners to interpret the correct meaning of their figurative use of language, they also want the listeners to know that they know they are deliberately being figurative. After all, if you say what is being false, people might lose confidence in the veracity of any of your statements. To prevent that from happening, you give slight hints that your language is figurative.

My take: the identification of sarcasm from words doesn't come from the words that are part of

the sarcastic statement, but from the introductory words that the speaker utters to indicate to the listener that sarcasm is coming up. Yet, these words are not mandatory, not normalised, not meant to be crystal clear, and not really necessary. We need a full "understanding" of the language in order to understand sarcasm.

2 Data

The data used here is described in (Khodak et al., 2018). It is scraped from reddit and is a subset of comments published between over a period between 2009 and 2017.

The labelling was done by the authors on the reddit platform themselves, so it tells whether commenters themselves see their text as sarcastic, rather than whether other people perceive it as such.

Several cleaning operations were carried out to create the original dataset: noisy and uninformative comments were removed, comments that were descendents of sarcastic comments were removed, URLs were removed, and non-ASCII characters were converted into ASCII.

The original dataset contained the whole sequence of a thread leading up to a sarcastic comment, but the presently used dataset contains only a single `parent_comment`. The dataset contains about nine hundred thousand labelled comments and a test set of about one hundred thousand unlabelled comments, totalling about one million records.

3 Choice of model

Given the general characteristics of the notion of sarcasm and the specific dataset for which we want to make predictions a choice of model should address the following conditions:

- Actual sarcastic statements can be made with the full variety of the language
- There are no definite social norms for introducing sarcasm.
- The actual dataset might contain non-grammatical language
- The actual dataset contains markup, which is not grammatical

These conditions are well-suited to be addressed by:

- Transformer architecture, as specified in (Vaswani et al., 2017)
- Pre-training on a large body of texts so the model has extensive "knowledge" of relations between words. E.g. BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).
- Byte Pair Encoding (BPE) tokenizer rather than traditional word-based tokenizer

Both BERT and RoBERTa fulfill the above conditions. RoBERTa is architecturally BERT, but with a different pre-training scheme. As RoBERTa performs better on multiple tests - as demonstrated in (Liu et al., 2019), here I selected RoBERTa.

4 System

Implementation was done in python using Simpletransformers[2] which is a layer over Hugging face transformers[3] Simpletransformers contains a standard implementation for binary classification of texts. Dataframes serve as input format for the training and evaluation functions of Simpletransformers. All that is needed is to given the columns to be read the names expected by Simpletransformers.

The underlying used model was `RoBERTa-base`[4].

Used GPU was an RTX 2070.

5 Data preparation

The original dataset was already cleaned of non-ASCII characters. All that needed to be done in addition to that was to read data from the CSV file such that relevant fields like `comment` and `parent_comment` were converted to strings and that any NaN values for these string fields were converted to empty strings.

RoBERTa's Byte Pair Encoding tokenizer elegantly automates any vocabulary operations traditionally needed for preparing textual data and it also deals elegantly with grammatical exuberance and with markdown[5].

[2] https://simpletransformers.ai/,

[3] https://huggingface.co/transformers/.

[4] https://github.com/pytorch/fairseq/tree/master/examples/roberta

[5] *"Byte Pair Encoding — The Dark Horse of Modern NLP"*, Akashdeep Singh Jaswal, 22 Nov 2019, https://towardsdatascience.com/byte-pair-encoding-the-dark-horse-\of-modern-nlp-eb36c7df4f10

6 Training and evaluation

As training with even a subset of the large dataset takes a long time, I first did several experiments with smaller subsets of some 10,000 items.

- Is there significant impact of markup?
- Is there significant impact of adding the `parent_comment` to the `comment`?[6]
- Is the estimated processing time for training, evaluation, and prediction making attaining the deadline impossible?

In all cases my experiments showed that it didn't.

From the training dataset I used 200,000 items for training, 10,000 for development (evaluation during training), and 20,000 for evaluation after the training was finished. An experiment with different sized training sets shows less than one percent improvement when training 200,000 instead of 100,000 items.

Training with 200,000 items takes about thirty minutes per epoch.

Training	Precision	Recall	F1
4,400	0.685	0.730	0.707
100,000	0.690	0.830	0.753
200,000	0.758	0.761	0.759

Table 1: Metrics per number of training items

7 Issues

I struggled with technical issues. While I have used Simpletransformers before with good results, I now found that:

- Early-stopping didn't work
- The selection of the best model after training did not in fact select the best model

Early stopping functionality could just be skipped, but not obtaining the best model was a show-stopper.

I found a workaround by parsing intermediate training results, converting them to the desired metrics, and and then selecting the model associated with the best metrics.

[6] In (A. and D., 2020) we find a slightly better result when training only on the sarcastic comment. But in (Dong et al., 2020) we find a significant (7%) improvement when the context is included.

8 Comparison with other work

A subset of 4400 items of the Khodak reddit dataset was used as a basis for a task of the second *Figurative Language Workshop.* In the overview paper of that workshop (Ghosh et al., 2020) we find a summary of an extensive mix of approaches[7]. Many of the considerations in the current approach can be found there.

Following the overachieving number one, the sub-top ranking efforts in that workshop got an F1-score of about 0.75. In 1 the F1-score for 4400 items was 0.707.

9 Improvements

The current results were attained without any hyperparameter optimisation. This takes a lot of processing time, but might lead to positive results.

Influence of author, score, and subreddit category have been ignored in the current approach. From (Khodak et al., 2018) we know that there is a correlation, e.g. more sarcasm in the `worldnews` subreddit than in `ask reddit`. But is there causation? Of course, such an improvement will be specific for the currently studied dataset and not lead to generally better detection of sarcasm.

Lastly, quite generally, looking at the results of different contests, we find that best-ranking teams have often applied ensemble methods and data augmentation. Both approaches should be possible in the area of sarcasm detection too.

10 Conclusion

This was rather fun. I didn't plan on getting into the topic as deeply as I did. It was never meant to be a research project. I enjoyed putting my existing skills in practice and I also enjoyed reading up on the efforts of others and better understanding the many ways in which solutions can be improved.

References

Kalaivani A. and Thenmozhi D. 2020. Sarcasm identification and detection in conversion context using BERT. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 72–76, Online. Association for Computational Linguistics.

Sourav Das and Anup Kumar Kolya. 2021. Parallel deep learning-driven sarcasm detection from pop

[7] If you are interested in the full proceedings of the workshop, you can find them at `https://aclanthology.org/2020.figlang-1.pdf`.

culture text and english humor literature. In *Proceedings of Research and Applications in Artificial Intelligence*, pages 63–73, Singapore. Springer Singapore.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Xiangjue Dong, Changmao Li, and Jinho D. Choi. 2020. Transformer-based context-aware sarcasm detection in conversation threads from social media. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 276–280, Online. Association for Computational Linguistics.

Debanjan Ghosh, Avijit Vajpayee, and Smaranda Muresan. 2020. A report on the 2020 sarcasm detection shared task. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 1–11, Online. Association for Computational Linguistics.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2018. A large self-annotated corpus for sarcasm. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.