Sarcasm Detection with Transformer Model Encoders

Emil Säll
Linköping University
emisa182@student.liu.se

Abstract

This paper approaches the problem of detecting sarcasm in a text. A text can be sarcastic on its own or given a context, both scenarios are considered and it can be seen as a binary classification problem. Sarcasm is something that could be hard to understand even in speech and it has been showed that humans often relies more on the intonation in a voice than the context to understand sarcasm. To approach a problem of sarcasm detection in text, where the intonation in a voice is non existent, yields a challenge where the text itself and a possible context is the only thing that can be relied upon.

The problem is approached by using the encoder part of transformer models. The data used consists of tweets, reddit comments and news headlines collected from multiple sarcasm-annotated datasets. During the project due to hardware limitations, the amount of data that could be used in the model was heavily reduced from about 1 million data points to about 50000. This of course effected the result and compared to the baseline which was trained on about 1.5 million reddit comments, the model presented in this paper was still performing similarly.

1 Introduction

The problem of detecting sarcasm can be difficult in an everyday life. It can depend on how well you know each other or in what context it is presented. To start with, the definition of sarcasm according to the Britannica dictionary is

> *the use of words that mean the opposite of what you really want to say especially in order to insult someone, to show irritation, or to be funny.* (Dictionary, 2023)

With this definition, the problem that is approached could be explained as finding the opposite meaning of a text when it is intended to be mean, funny or to show annoyance. It is a binary classification problem.

When exploring the topic of detecting sarcasm in connection to this paper, it is implied that the detection of sarcasm is done by a computer. A perspective that could help understanding the depth of the problem is to consider how humans learn sarcasm. This is something that is done in the early years in life and in the paper *How Children Understand Sarcasm: The Role of Context and Intonation* released in 1990 it was concluded that children relies more heavily on the intonation to understand sarcasm than the context given. (Capelli et al., 1990) This has been taken further where, for instance, the article *The Sound of Sarcasm* (Henry S. Cheang, 2007) from 2007 was presented, where the phonetics of sarcasm were analysed. This focused on the acoustic cues in speech and was successful in finding some characterisation to detect sarcasm. When it comes to detecting sarcasm in text, the absence of intonation in a voice leaves the context as the primary cue for understanding and that is the problem that is being approached in this paper. It can be explained by a conditional probability

$$\mathrm{P}(text = sarcasm \quad | \quad context). \qquad (1)$$

2 Theory

To detect sarcasm in a text is a problem that has been approached in the latest years with multiple techniques. A few of these are covered in the paper *Techniques of Sarcasm Detection: A Review* (Verma et al., 2021). In this paper the focus is on transformer models and will come back to the comparison of other models later on.

To understand where the root of transformer models began and what they are capable of, the starting point can be set to the year 2014. This year

the paper *Neural Machine Translation by Jointly Learning to Align and Translate* (Bahdanau et al., 2014) was published and then presented at the International Conference on Learning Representations (ICLR) in 2015 which approached the problem of translation with neural networks. The paper introduced what now is known as *Attention* which is a mechanism that uses probabilities or weights to acknowledge the relevance in each part of the input. The weights can be interpreted as where to focus your attention in the text and hence the name was formed. This relevance acknowledging feature turned out to yield great results in language translation.

The attention mechanism which is used today consists of three major parts: queries Q, keys K and values V. Each query is connected to a key and it is used to obtain information of each key-value pair which is some numerical representation of the input. When looking at the equations from Bahdanau, the values and keys were the same. The Attention itself which can be seen as a context vector is calculated with

$$\text{Attention}(Q, K, V) = \sum_i \alpha_{Q,K_i} V_{K_i} \quad (2)$$

where V_{K_i} is the value connected to the key K_i. The term α_{Q,K_i} could be seen as the attention weights and is calculated through the softmax function

$$\alpha_{Q,K_i} = \text{softmax}(e_{Q,K_i}) = \frac{\exp\left(e_{Q,K_i}\right)}{\sum_j \exp\left(e_{Q,K_j}\right)}. \quad (3)$$

In the softmax function there is a term e_{Q,K_i} which could be calculated with an alignment model a,

$$e_{Q,K_i} = \text{a}(Q, K_i). \quad (4)$$

The alignment model can vary, but in the first paper a perceptron was used. In Figure 1 the attention mechanism is visualized. This mechanism is the base on which transformer models is created upon.

During the year 2017 the paper *All you need is Attention* (Vaswani et al., 2017) introduced transformer models. Transformer models are based on Attention and performs the Attention mechanism in parallel yielding a rather complex

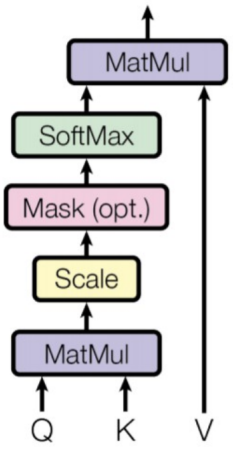

Figure 1: Visualization of the Attention mechanism. (Vaswani et al., 2017)

but efficient structure. This parallel process is called *Multi-Headed Attention* and is visualized in Figure 2.

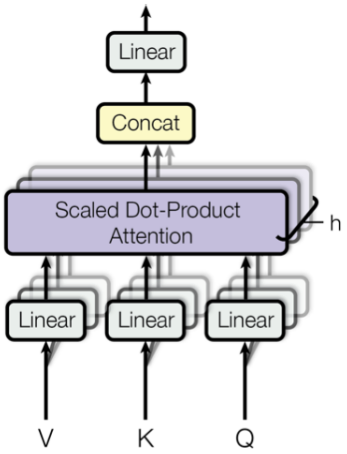

Figure 2: Visualization of Multi-headed Attention. (Vaswani et al., 2017)

In the multi-headed attention the mechanism that is used is called *Scaled-Dot-Product Attention* which is calculated through

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \quad (5)$$

which is the dot product of the queries and the keys divided by the square root of the individual key dimension d_k that provides the scaling. The strength of using the multi-headed attention apart from the parallelization, is that each attention head will learn differently which yields in both an

effective and a versatile model. (Vaswani et al., 2017)

The Transformer model consists of an Encoder and a Decoder. The Encoder, visualized in the left side of Figure 3, consists of the multi-headed attention together with a feed forward network. Before the data is ready to be used in the transformer model it needs to be preprocessed which is described in a further extent in the chapter 3 Data. The preprocessing makes embeddings of the input which is a process that could include tokenization or lemmatization to mention a few methods. After these methods a numerical representation of the input is created which is referred as an embedding. With the positional encoded numerical representations, the data is ready for the Encoder.

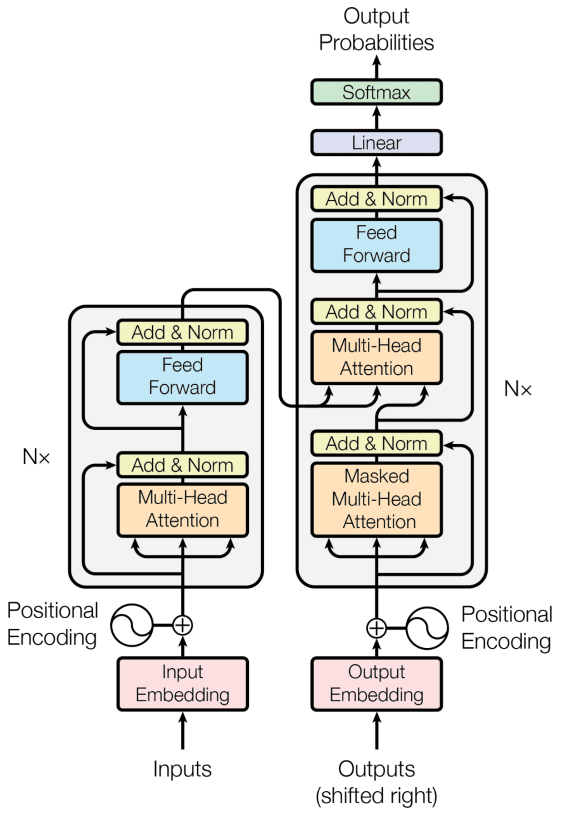

Figure 3: Visualization of the transformer architecture. (Vaswani et al., 2017)

The Decoder, visualized in the right side of Figure 3, uses the multi-headed attention mechanism in two steps. A masked multi-headed attention which is implemented to ensure that the model only bases its training on previously seen data. The main multi-headed attention in the decoder uses the outputs from the encoder as the values and keys and the output of the masked multi-headed attention as the queries. In this way the decoder can focus the attention on the important parts of the new inputs since it has learned from previous data. With this structure, the transformer model has been successful in tasks that uses sequential data which texts can be seen as. (Vaswani et al., 2017)

3 Data

The data used in the project consists of tweets, reddit comments and news headlines collected from pre-existing datasets. (debanjanghosh, 2020; Khotijah et al., 2020; Misra and Arora, 2023; SAR, 2017) It is notable that these texts are often quite short which will influence the model. In the pre-existing datasets all the target texts are annotated to be sarcastic or not. Some of the data have a context connected to target text to give further information of the scenario where or how the target text appeared. Since not all of the data comes with a context, the problem of sarcasm detection is approached with both a context and not.

The majority of the pre-existing datasets consists of more types of data than the data which is interesting for this project. The interesting data is the target text, the sarcasm label and the context if it exists. Because of this, a new dataset was created where the data from the pre-existing datasets were restructured and relabeled to get consistency through all of the data. The new dataset was created in the structure showed below with the columns `text`, `label` and `context`.

```
text                    label
The target text         0: Not sarcastic
                        1: Sarcastic

-----------------------------------------
context
NaN: If no context is provided.
The context if provided.
```

Since the data is collected and merged from pre-existing datasets it is essential for the model performance to check whether the data is skewed or not. In Figure 4 it can be seen that the 1085507 data points is fairly evenly distributed. This minimizes the risk of the model being affected of skewed data.

Further preprocessing of the data is described in

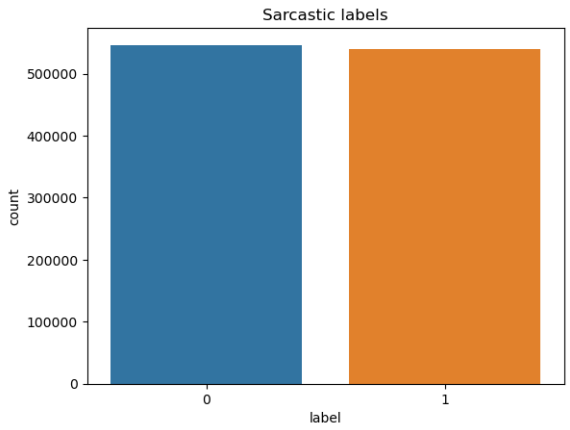

Figure 4: The distribution of the labeled boolean data.

the 4 Method chapter.

4 Method

With the data ready in a `Pandas DataFrame` with

text	label	context

as the column names, the target is to get a numerical representation of the data that can be passed into a transformer model. As previously stated, the detection of sarcasm in text relies on the context that is given. This leads up to the decision of using contextualized word embeddings. In contextual word embeddings, the words connected to the target word is taken into account in the numerical representation. This yields that the embedding for example the word *watch* will differ if it comes to a context where it means to watch something or if it means a watch like a clock.

To implement the contextual word embeddings the language pipeline `en_core_web_trf`, v.3.7.3, from `spaCy` is used. It is a pipeline that is created based on the language model RoBERTa, Robustly optimized BERT approach, (Liu et al., 2019) which was created due to the realization that BERT, Bidirectional Encoder Representations from Transformers, (Devlin et al., 2019) was undertrained. This pipeline performs several actions on the input text to be able to create a numerical interpretation. The main ones are described below. (Explosion, 2023)

- **Transformer:** It uses the RoBERTa transformer model which tokenizes the data and enables the contextual word embeddings.
- **Tagger:** Every token gets tagged with the part-of-speech tag which corresponds to what category grammatically the word belongs to.
- **Parser:** The parser connects the grammatical dependencies throughout the text and assigns each token its role.
- **Lemmatizer:** The tokens get lemmatized.
- **Named Entity Recognizer:** The tokens will be classified as a certain entity which could be `DATE` or `PERSON` for an instance. If a token does not fit into any of the entities it will be left empty.

Since the majority of the data consists of public posts from social platforms some of the used words might not be recognized by the language pipeline `en_core_web_trf`. There could be misspelling, slang or some other words not recognized and because of this a custom component of the language pipeline is implemented called `tensor2attr`. (Hiippala, 2020) The custom component enables each input text to be represented as one single embedding to compare with the more classic approach of using word embeddings, an embedding for each word. With this feature the modelling of $\mathrm{P}(text = sarcasm \mid context)$ will have a more intuitive structure where there will be one embedding for the context and one embedding for the text.

With the described methods above, the data is preprocessed to consist of the embedding for each text and context together with the corresponding labels whether the original text was sarcastic or not. This makes the data ready for the transformer model.

This model for sarcasm detection is implemented with the `PyTorch` `nn.TransformerEncoder` together with `nn.TransformerEncoderLayer`. Since the problem that is being approached with the prepared data is a binary classification problem, the implementation of the transformer will only include the encoder. When the output only needs to be binary, there is not much to decode.

Finally the components for training the model is needed. An optimizer is used to update the parameters in the training process and here the decision

of optimizer became Adam, Adaptive Moment Estimation. Adam provides an adaptability that has proven to be beneficial for attention models such as transformers. (Zhang et al., 2020) Another part of the training process is the scheduler which adapts the learning rate depending on where in the training the model is. The scheduler used is the `optim.lr_scheduler.ReduceLROnPlateau` which reduces the learning rate when the model improvement decays. (PyTorchContributors, 2023)

The last part is the loss function which was set to a binary cross entropy, BCE, loss. It is a binary classification problem and with that, a binary loss function was evident. The BCE-loss is calculated through

$$-(y \cdot \log(p) + (1 - y) \cdot \log(1 - p)) \quad (6)$$

where y is the labeled data and p is the predicted probability. It will punish the predictions that are confident and at the same time wrong. (Leikin, 2023) During the training a 5-fold cross validation was used with the training data.

5 Results

When discussing the results some issues needs to be disclosed first. When the model was about to be trained and tested it became apparent that the amount of data prepared for the project was way over the laptop's limit. After the preprocessing when the representation of data consisted of the embeddings, the file would have needed about 30 GB of space. Unfortunately this needed resources that was not available. After some testing the limit of what the laptop could handle was about 70000 entries instead of the intended 1085507, a significant difference. The 70000 entries were sampled and then undersampled to get a balanced dataset and that is what these results are based on. After the undersampling, the dataset is balanced with 57080 entries in total yielding in 28540 samples of each label which is showed in Figure 5.
Before the training of the model started, the hyperparameters was set as showed below.

```
Size of training data = 80%
Size of testing data = 20%
Number of heads for the Multi-Headed
    Attention = 12
Number of layers = 4
Dropout = 0.1
```

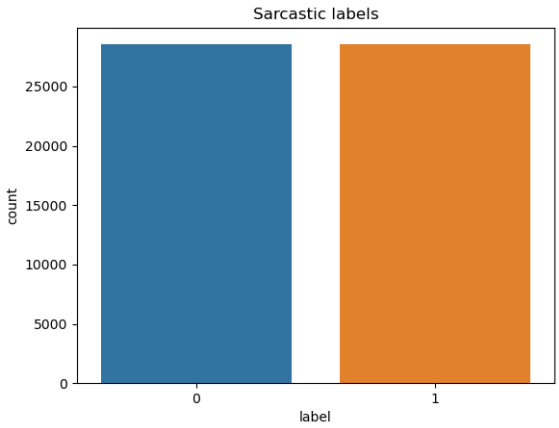

Figure 5: The distribution of the undersampled data.

```
Number of epochs = 5
Learning rate (or step size) = 0.0001
Weight decay = 0.0001
Batch size = 64
Number of folds for cross-validation = 5
```

These values were chosen by testing the model on a smaller set of the data. During the training, data was collected throughout the folds to see how the model performed. This data is presented in Table 1.

After the model was trained, the model was evaluated with unseen testing data. The results for the testing data is showed below.

```
Testing Accuracy:    0.6491
Testing Precision:   0.6289
Testing Recall:      0.7423
Testing F1-score:    0.6809
```

6 Discussion

Due to the great limitation of hardware that unfortunately became apparent when the model was next to complete, the results are not pleasing. There has been a lot of work put in to the model and to have this as a final result is disappointing. The initial idea of trying to identify sarcasm is still interesting. If more time could be put into the project it would be interesting to explore how to make the model better and how to enable the use of the whole dataset.

As previously mentioned, the paper *Techniques of Sarcasm Detection: A Review* (Verma et al., 2021) covers several methods of sarcasm detection where the Soft-Attention function based Bi-

Measure	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	**Average**
Accuracy	0.6359	0.6250	0.6376	0.6436	0.6501	**0.6384**
Precision	0.7296	0.5844	0.6052	0.7007	0.6273	**0.6495**
Recall	0.4167	0.8354	0.7886	0.5144	0.7467	**0.6604**
F1-Score	0.5304	0.6878	0.6849	0.5933	0.6818	**0.6356**

Table 1: The validation results of the model.

directional long short-term memory by convolution networks showed great result with an accuracy of 93.71% for a random tweet dataset. This is a result that outperforms this model without a doubt. Even if the model presented in this paper uses rather complex models, the structure is pretty simple. An overview of the whole process is visualized in Figure 6.

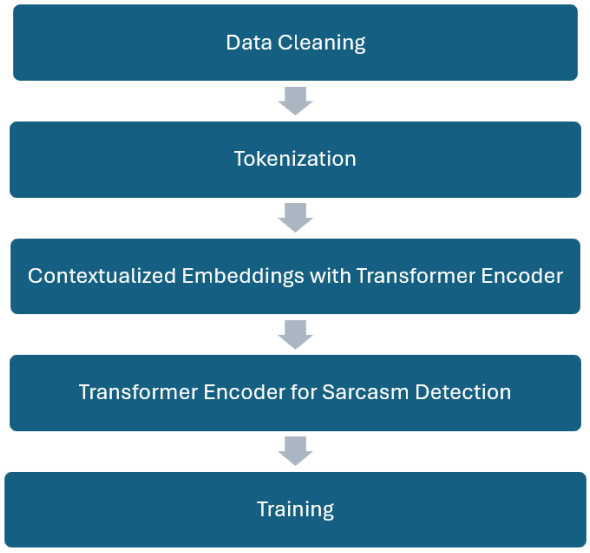

Figure 6: An overview of the process.

To use the best model found as a baseline is a hard match. If you google *sarcasm detection transformer*, the first article that appears is *Transformers on Sarcasm Detection with Context*. (Kumar and Anand, 2020) In this paper multiple models have been tested on twitter and reddit comments, all with related context. The results from reddit data and twitter data are separated when presented and if a baseline-comparison should be chosen the decision will be on the reddit results. In the merged data for this project, there is a clear majority of data from reddit compared to twitter. This leads to the reddit models being a better comparison. In Figure 7 the results from this paper is showed. In the paper, multiple models are tested and the model that should compare the most to the one created for this is the `roberta-large`$_{SS}$. The base of model used in this paper have also used roberta and the SS stands for Single Sentence classification which has got the most similar structure to this model where the single target text gets connected to the single context. This gives the comparison results showed below which unfortunately does not include the accuracy.

```
Precision: 0.675
Recall:    0.675
F1-Score:  0.6749
```

The dataset used in the baseline consisted of 1.5 million texts, which is about 30 times bigger than the dataset used for the model that in the end was trained.

Model	**P**	**R**	**f1**
response only$_{SS}$	64.2	64.7	63.83
bert-base$_{SS}$	66.5	66.6	66.47
bert-large$_{SS}$	67.3	67.3	67.27
roberta-large$_{SS}$	67.5	67.5	67.49
spanbert-base$_{SS}$	66.9	67.3	66.75
spanbert-large$_{SS}$	67.4	67.4	67.36
bert-large$_{LoT}$	68.1	68.1	68.0
roberta-large$_{LoT}$	69.3	69.9	**69.11**
roberta-large$_{ST}$	67.9	68.1	67.86
roberta-large$_{DT}$	68.1	68.1	68.1

Figure 7: The results from the chosen baseline.

If the results of the two models were to be compared, the model presented in this paper could be deemed to perform slightly better than the baseline seen to the F1-score, but very similar. Since the F1-score is the harmonic mean of the recall and the precision it is a good measure in a binary classification task. With that said, the presented model has been trained on significantly less data than the baseline. This could yield that the model is not as robust as the baseline, making the baseline a safer bet to use.

7 Conclusion

The problem was approached with high ambition and even though the issues regarding data size lead to a model that did not perform as well as thought, there is still potential in this method. To make this deep dive into transformer models has been very time consuming, but it has rewarded me with a much greater understanding of these models and the complexity behind them. The problem of detecting sarcasm could have been solved with better results than the ones presented in this paper. The interest of making this project was to explore the concept of transformer models and the attention mechanism and it has yielded in a great result with that in mind.

The results are similar to the baseline. Considering that the presented model is trained with significantly less data than the baseline, these results are promising if the model would be trained on the whole dataset.

References

2017. A large self-annotated corpus for sarcasm.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate.

Carol A. Capelli, Noreen Nakagawa, and Cary M. Madden. 1990. How children understand sarcasm: The role of context and intonation. *Child Development*, 61:1824–1841.

debanjanghosh. 2020. Educationaltestingservice - sarcasm. https://github.com/EducationalTestingService/sarcasm.

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

The Britannica Dictionary. 2023. sarcasm. https://www.britannica.com/dictionary/sarcasm.

Explosion. 2023. en_core_web_trf-3.7.3. https://github.com/explosion/spacy-models/releases/tag/en_core_web_trf-3.7.3.

Marc D. Pell Henry S. Cheang. 2007. The sound of sarcasm. *Speech Communication*, 50.

Toumo Hiippala. 2020. Word embeddings in spacy.

Siti Khotijah, Jimmy Tirtawangsa, and Arie A. Suryani. 2020. Using lstm for context based approach of sarcasm detection in twitter. IAIT2020, New York, NY, USA. Association for Computing Machinery.

Amardeep Kumar and Vivek Anand. 2020. Transformers on sarcasm detection with context. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 88–92, Online. Association for Computational Linguistics.

Igal Leikin. 2023. Understanding binary cross-entropy and log loss for effective model monitoring.

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.

Rishabh Misra and Prahal Arora. 2023. Sarcasm detection using news headlines dataset. *AI Open*, 4:13–18.

PyTorchContributors. 2023. Reducelronplateau.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Palak Verma, Neha Shukla, and A.P. Shukla. 2021. Techniques of sarcasm detection. pages 968–972, Galgotias College of Engineering and Technology, Gr. Noide, India. International Conference on Advanced Computing and Innovative Technologies in Engineering (ICACITE), Springer.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, and Suvrit Sra. 2020. Why {adam} beats {sgd} for attention models.

A Declaration of use of generative AI

During the project the generative language model ChatGPT has been used to provide snippets of code. The version that has been used is 3.5 which is based on data from September 2021 and earlier. The model of version 3.5 does not have access to data produced after that. Because of the subject area of transformers it became clear that ChatGPT 3.5 did have some knowledge of transformers, but not to the extent I hoped for. It always has suggestions of how to proceed, but these rarely works. Because of this, the major thing ChatGPT provided the project with was inspiration to the code structure which has been used.