# Sarcasm Detection
## 3-cfu Project Work

**Yuwei Ke**
Master's Degree in Artificial Intelligence, University of Bologna
yuwei.ke@studio.unibo.it

## Abstract

Sarcasm detection is a classic natural language processing task. Sarcasm is a common pragmatic phenomenon in daily communication that can enrich the speaker's point of view and indirectly express the speaker's deep meaning. The research goal of the sarcasm detection task is to dig into the irony tendency of the target statement. In this project, I use natural language processing techniques to detect sarcasm in the Reddit dataset. The main approach is based on the paper "A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks" (Poria et al., 2016). I investigate the performance of three distinct models (CNN, LSTM, and Bert) and compare their results. The best one is selected as the baseline along with the sentiment, emotion, and personality features extracted by the pre-trained models to detect sarcasm. Bert, unsurprisingly, gives the best result in this task. But after adding the extracted features into the model, it doesn't achieve a better performance. This shows the strong capability of the Bert model in sarcasm detection tasks and not all the models can improve performance by adding such features.

## 1 Introduction

Sarcasm is a common linguistic phenomenon, usually using metaphor, exaggeration, and other techniques to expose, criticize or ridicule someone or something. There may be a difference between the literal and implied meaning of a sentence that contains sarcasm. For example, the real meaning of the sentence 'I absolutely love to be ignored !' is that I don't want to be ignored. As social media platforms quickly gain popularity, an increasing number of Internet users are publishing their ideas and opinions online, including plenty of sarcastic expressions. So, being able to recognize sarcasm will be helpful. For example, if sarcastic comments about a product can be detected, the company can better understand the feelings and feedback of users on the product. But it may be challenging to identify sarcasm for the distinction in the meaning.

The methods used to detect sarcasm are constantly being updated. Machine learning techniques like SVM (Cortes and Vapnik, 2004), random forest (Ho, 1995), decision trees (Quinlan, 1986), etc. were previously used to identify sarcasm. Then is followed by deep learning methods such as CNN(Convolutional Neural Network) (Lecun et al., 1998) and LSTM(Long short-term memory) (Hochreiter and Schmidhuber, 1997). After the emergence of Transformers (Vaswani et al., 2017), pre-trained models such as BERT (Devlin et al., 2018) and Roberta (Liu et al., 2019) have been applied in this field. The popular way now is multimodal combined with deep learning, using text, image, sound, and other information to do the detection.

In this project, I compare the performance of several previously popular deep learning methods in sarcasm detection and select the one that performs best. The CNN model is used as the baseline in the paper "A Deeper Look at Sarcastic Tweets Using Deep Convolutional Neural Networks" (Poria et al., 2016) and three different pre-trained models are used to extract sentiment, emotion, and personality features from the data. The performance of the CNN model is then enhanced by combining them with the detected feature of the baseline model. I carry it out in the same approach with better models to explore if it can make use of such information to improve classification performance.

First of all, I used Twitter Sentiment Analysis Dataset decsribed in 3.2 , Emotion Dataset in 3.3, and personality Dataset in 3.4 respectively to train three models for information extraction. The model architecture comes from this paper (Poria et al., 2016). I then tested the sarcasm of the Reddit dataset (3.1) using CNN, LSTM, and Bert. Although it took a long time to train Bert, it got the best results. Finally, I combined the Bert model with the extracted information to detect sarcasm.

## 2 System description

### 2.1 Feature extraction models

Sarcasm detection may not be able to understand the true meaning of sentences and conduct the proper classification if it is just approached as a text classification problem. According to the paper (Riloff et al., 2013), sarcasm in a statement may have a connection with sentiment or other aspects. Based on the models suggested in the paper, 3 different types of models are trained for sarcasm detection. They will extract sentiment, emotion, and personality features from the provided data.

#### 2.1.1 Sentiment model

Sentiment analysis is the process of detecting if a text has a tendency to be positive or negative. It involves analyzing, processing, concluding, and reasoning the subjective emotional text. The CNN architecture used in the paper (Poria et al., 2016) provides the basis for the sentiment model, which is trained using the Twitter sentiment dataset that was presented in section 3.2. The model is composed of Convolutional layer, Maxpooling layer, Dense layer, and Dropout layer. The detailed information is shown in Figure 1. The extracted feature from the fully connected dense layer has a length of 100 and it will be used to help detect sarcasm.

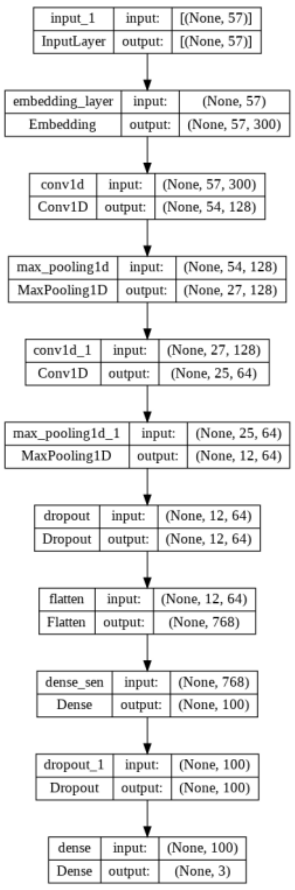
input_1 | InputLayer | input: [(None, 57)] | output: [(None, 57)]
embedding_layer | Embedding | input: (None, 57) | output: (None, 57, 300)
conv1d | Conv1D | input: (None, 57, 300) | output: (None, 54, 128)
max_pooling1d | MaxPooling1D | input: (None, 54, 128) | output: (None, 27, 128)
conv1d_1 | Conv1D | input: (None, 27, 128) | output: (None, 25, 64)
max_pooling1d_1 | MaxPooling1D | input: (None, 25, 64) | output: (None, 12, 64)
dropout | Dropout | input: (None, 12, 64) | output: (None, 12, 64)
flatten | Flatten | input: (None, 12, 64) | output: (None, 768)
dense_sen | Dense | input: (None, 768) | output: (None, 100)
dropout_1 | Dropout | input: (None, 100) | output: (None, 100)
dense | Dense | input: (None, 100) | output: (None, 3)

Figure 1: Sentiment Model architecture

#### 2.1.2 Emotion model

Emotion analysis is the process of identifying and analyzing the underlying emotions in textual data. To obtain the emotion feature in sarcastic text, the model is trained on the Emotion dataset described in 3.3. It's challenging to identify all of the many emotions that humans experience. The dataset only contains 6 types of emotions. The last output layer differs from the sentiment model's design in the model for emotion extraction, and the others are same. The output dense layer has 6 units with softmax activation to determine which emotion it is of the data. The extracted feature from the first dense layer has a length of 100 and it will be concatenated with other features to detect sarcasm in the final model.

#### 2.1.3 Personality model

Personality is the combination of an individual's behavior, emotion, motivation, and characteristics of their thought patterns (Mehta et al., 2020). Since the 1990s, the big five personality traits have been employed to categorize people's characteristics. The personality models are trained using the dataset mentioned in 3.4 in order to identify the personality within the five qualities of the comment author. Every data in the dataset matches more than one label. For better performance, 5 identical CNN models are used to perform the binary classification task for each type of personality. The model has some differences from what is described in the Sentiment model. The output space dimensionality of the first and second convolutional layers, respectively, is 600 and 300. The output of the first dense layer is a 100 dimensional feature vector and it will be used with other features to detect sarcasm in the final model.

### 2.2 Convolutional neural network

Convolutional neural network (Lecun et al., 1998) (CNN) is a special type of neural network, which performs particularly well on images. One or more convolutional layers make up its structure, which can lower the deep network's memory consumption. Through local connections and shared weights, it decreases the number of parameters, which makes the model simpler to train and less prone to overfitting. In this project, I employ a CNN architecture to recognize sarcasm, which is based on the design suggested in the paper (Poria et al., 2016). Two Convolutional layers and two Maxpooling layer make up the CNN model. A Flatten layer, a Dense layer with Relu activation, and a Dropout layer are present after the second Maxpooling layer. The last output layer is a Dense layer with 1 unit and Sigmoid activation. Figure 2 displays the specific model in detail.

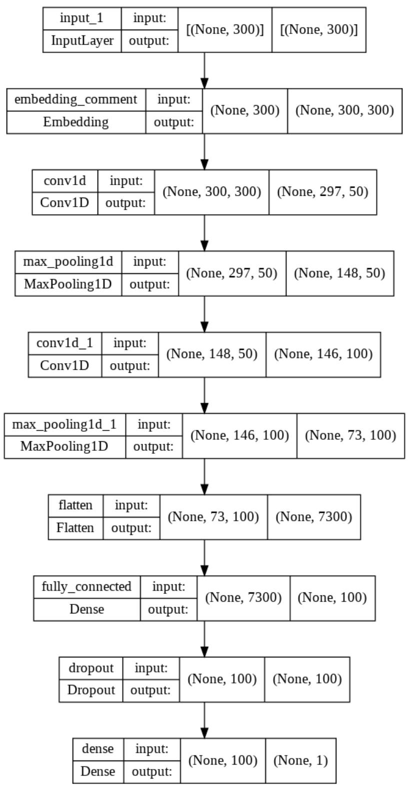
input_1 | InputLayer | input: [(None, 300)] | output: [(None, 300)]
embedding_comment | Embedding | input: (None, 300) | output: (None, 300, 300)
conv1d | Conv1D | input: (None, 300, 300) | output: (None, 297, 50)
max_pooling1d | MaxPooling1D | input: (None, 297, 50) | output: (None, 148, 50)
conv1d_1 | Conv1D | input: (None, 148, 50) | output: (None, 146, 100)
max_pooling1d_1 | MaxPooling1D | input: (None, 146, 100) | output: (None, 73, 100)
flatten | Flatten | input: (None, 73, 100) | output: (None, 7300)
fully_connected | Dense | input: (None, 7300) | output: (None, 100)
dropout | Dropout | input: (None, 100) | output: (None, 100)
dense | Dense | input: (None, 100) | output: (None, 1)

Figure 2: CNN Model architecture

### 2.3 LSTM

Long short-term memory, or LSTM (Hochreiter and Schmidhuber, 1997), is a kind of RNN (Recurrent Neural Networks) which can solve the shortage of short-term memory of RNN. It is challenging for RNN to transmit information from the earlier time step to the later time step when a sequence is lengthy enough. However, the LSTM can link the context because it can learn long-term dependant information and recall the data from a previous time step. To complete the goal for this project, I utilize a simple LSTM model. It is composed of one bidirectional LSTM layer with 128 units, a Dense layer with 128 units, and a Dropout layer to prevent overfitting. The last output layer is a Dense layer with 1 unit and Sigmoid activation. The detailed model information is shown in Figure 3.

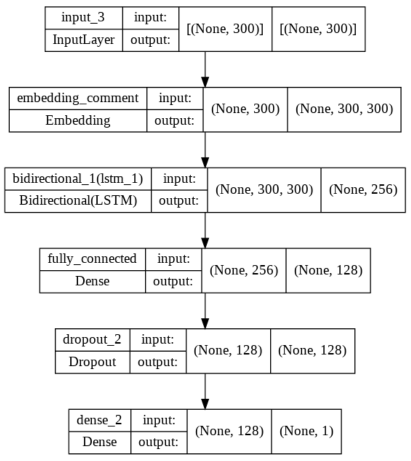
input_3 | InputLayer | input: [(None, 300)] | output: [(None, 300)]
embedding_comment | Embedding | input: (None, 300) | output: (None, 300, 300)
bidirectional_1(lstm_1) | Bidirectional(LSTM) | input: (None, 300, 300) | output: (None, 256)
fully_connected | Dense | input: (None, 256) | output: (None, 128)
dropout_2 | Dropout | input: (None, 128) | output: (None, 128)
dense_2 | Dense | input: (None, 128) | output: (None, 1)

Figure 3: LSTM Model architecture

### 2.4 Bert

In 2018, Google developed and unveiled the deep learning model BERT (Devlin et al., 2018) (Bidirectional Encoder Representation from Transformers). It is a model that has already been trained and may be applied immediately to other NLP tasks. It proved consistently better than other language models proposed at the time and performed remarkably well for many other tasks later on. I'm using the uncased bert model and corresponding text processor in this project which is provided Tensorflow Hub. The Bert model has 12 hidden layer, a hidden size of 768 and 12 attention heads for each layer. The Bert's pooled output feature will be placed into the Dropout layer with drop rate of 0.2. The final output layer is a Dense layer with 1 unit and Sigmoid activation. The sequence length of the BERT model is 128. The detailed model information is shown in Figure 4.

## 3 Data

### 3.1 Sarcasm Dataset

This is a corpus conducted by (Khodak et al., 2017) in the paper "A Large Self-Annotated Corpus for Sarcasm". The data is in millions, which solves the previous problems of the sarcasm corpus being small in scale, narrow in scope, or undivided in the field. In addition, the data set contains contextual information such as the user name, the forum topic

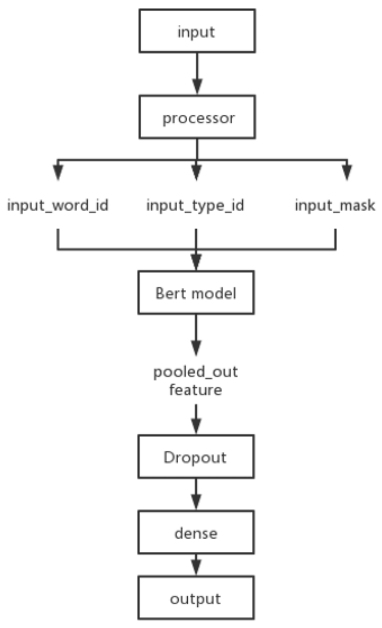

Figure 4: Bert Model architecture

to which the sentence belongs, the response to the sentence, and so on.
I downloaded the dataset from Kaggle, and it contains 3 data files: train-balanced-sarcasm.csv, test-unbalanced.csv, and test-balanced.csv. The link for dataset is in 7. In this project, I only use the train-balanced-sarcasm.csv file. It has 1010,826 pieces of data. Because a large amount of data requires a high level of operation configuration, I selected 100,000 pieces of data, 50000 sarcastic and 50000 non-sarcastic, and split them into train, verification, and test data sets in the ratio of 8:1:1. Only the fields "comment" and "label" are used in the project. Label 0 represents non-sarcastic data, while label 1 means sarcastic data.
The data has been pre-downloaded and placed in a Pandas dataframe, and carefully preprocessed, including the removal of special and unknown symbols, and HTML links. The sentences have all been changed to lowercase. The stopwords were not removed because I tested them both with and without removal and found that the accuracy was higher when the stopwords were kept. This might be as a result of the fact that the stopwords also provide sarcastic semantic information. For CNN and LSTM model, the data should be converted into word embeddings prior to training. A vocabulary is initially constructed. The GloVe 6B 300d pre-trained word vectors are downloaded using Gensim, and each word in the vocabulary is given a distinct number using Keras Tokenizer. <OOV> is used as a token parameter for out-of-vocabulary terms. An embedding matrix is built and OOVs are handled by assigning a random embedding and selecting values from a uniform distribution. Each sentence in dataset are post-padded with zeroes up to 300. Comments are shown as a list of integer tokens.
For Bert, the data is preprocessed by the bert uncased preprocessor, and then is transformed into fixed-length input sequence. The sequence consists of a tensor input_word_ids with numerical ids for each tokenized input, including start, end and padding tokens, plus two auxiliary tensors: an input_mask (that tells non-padding from padding tokens) and input_type_ids for each token (that can distinguish multiple text segments per input).

## 3.2 Sentiment Dataset

To train a sentiment model for extracting features from the sarcastic dataset. For training, I utilize Twitter Sentiment Analysis Dataset from Kaggle. The link for dataset is at 7. It has 162980 tweets in all; 162969 remain after the null value data has been removed. There are 72249 positive data, 55211 neural data, and 35509 negative data among them. The data are processed by removing the special characters, uncommon symbols, links, and stopwords.

## 3.3 Emotion Dataset

To train an emotion model for extracting features from the sarcastic dataset. I use the Emotion Dataset for NLP from Kaggle. The link for dataset is at 7. Three sections of the dataset - train, test, and validation - each containing 16,000, 2,000, and 2,000 pieces of data, respectively, have already been created. Six categories are included on the label: love, joy, sadness, anger, fear, and surprise. Joy has 6761 instances, sadness has 5797, anger has 2709, fear has 2373, love has 1641, and surprise has 719. During the processing period, just the stopwords are eliminated because the text is clean.

## 3.4 Personality Dataset

To train the personality model for extracting features from the sarcastic dataset, I use a large dataset of 2400 stream-of-consciousness texts annotated with a binary label of the Big Five personality. The link for dataset is at 7. It is developed by Pennebaker and King in 1999 and gathered using a standardized self-report questionnaire. The Big

Five Personality traits are Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism. Each essay may be labeled with more than one personality. The data are cleaned up by eliminating stopwords, unusual symbols, and special characters.

## 4 Experimental setup and results

### 4.1 setup

This project was implemented at Google Colab environment. Some libraries used in the project are as shown in Table 1:

| | |
|---|---|
| Pandas 1.3.5 | Numpy 1.21.6 |
| Re 2.2.1 | tensorflow_hub 0.12.0 |
| Gensim 4.0.0 | Tensorflow 2.8.4 |
| Keras 2.8.0 | tensorflow-text 2.8.1 |
| Nltk 3.8.1 | tf-models-official 2.7.0 |

Table 1: libraries used

### 4.2 Feature extraction models

During training, Adam optimizer is used, and the batch size is 64. By using ModelCheckpoint, the model with minimum val loss will be saved into an H5 file. The sentiment model uses a learning rate of 1e-5 to train for 40 epochs. The loss function is categorical cross-entropy and it gets an accuracy of 89% and f1 score of 88% on the test set. The emotion model uses a learning rate of 5e-5 to train for 40 epochs. The loss function is categorical cross-entropy and it gets an accuracy of 89% and f1 score of 85% on the test set. The personality models use a learning rate of 1e-4 to train for 10 epochs. The loss function is binary cross-entropy and the val accuracy for each model is shown in Table 2.

| | val_accuracy |
|---|---|
| cEXT | 0.5480 |
| cNEU | 0.5560 |
| cAGR | 0.5480 |
| cCON | 0.5840 |
| cOPN | 0.6360 |

Table 2: Best val accuracy of 5 personality models

### 4.3 Baseline selection

- CNN model: during training, it uses Adam optimizer and binary cross-entropy loss. The learning rate is 5e-5 and the batch size is 32. Each epoch lasts around 32 seconds and it will be trained for 40 epochs. The EarlyStopping method monitors the minimum val loss and the training will stop if there is no improvement in the val loss in 10 epochs.

- LSTM model: the learning rate for this model is 1e-4. Other training configurations are the same as that in the CNN model. It takes around 102 seconds for each epoch.

- Bert model: the learning rate for this model is 2e-5, the optimizer is Adamw and the loss is binary cross-entropy. The batch size is 16 and it will be trained for 10 epochs. Every epoch takes around 26 minutes to train.

The result for these three models is shown at Table 3.

| | accuracy | recall | f1 score |
|---|---|---|---|
| CNN | 0.67 | 0.67 | 0.66 |
| LSTM | 0.69 | 0.69 | 0.69 |
| Bert | 0.72 | 0.72 | 0.72 |

Table 3: Performance of CNN, LSTM, Bert

### 4.4 Final model

We can conclude from the above results that the performance of Bert is better than that of other models, so I choose Bert as the baseline of this task, and combine the features extracted from the three pre-training models to form the final model. The model input passes through the preprocess KerasLayer and then through the pre-trained Bert model provided by the TensorFlow hub. The 'pooled_output' of the result will be concatenated with the predicted feature of sentiment, emotion, and personality pre-trained models. A Dropout layer with a 0.2 drop rate will then be applied to them. The final output layer is a dense layer with 1 unit and sigmoid activation. The arthitecture is shown at Figure 5. The model is trained for 10 epochs and the learning rate is 2e-5. The Adamw optimizer and binary cross-entropy are employed during the training. It takes around 41 minutes to train 1 epoch. It gets a result of 0.7155 accuracy and 0.72 f1 score. The experimental results are not as good as expected. The final model with the three types of features added get similar results to the model without the addition. Since the pre-trained personality models' performance is not good and the accuracy rate is relatively low. Therefore, I decided to remove the personality feature from the model, but only with sentiment and emotion, and then experimented again. It gained a 0.7221 accuracy and 0.72 f1 score. The model score basically doesn't change.

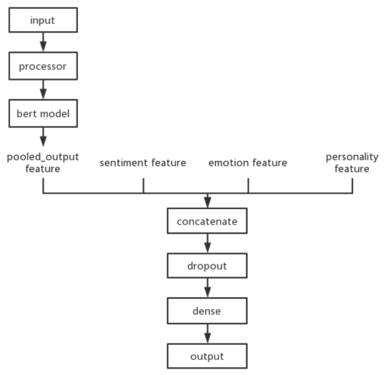

Figure 5: Final Model architecture

## 5 Discussion

In this project, according to the experimental process, steps and results in the paper (Poria et al., 2016), adding sentiment, emotion, and personality features to the model to detect sarcasm can improve the performance of the model and promote the accuracy of detection. But in my experiment, it didn't show the anticipated effect. And the models didn't produce very good outcomes. I believe the following are the possible factors:
Firstly, the dataset includes around 1.3 million data but I only use 100,000 for the limit of recourse. The first epoch's val accuracy was 6% higher than the best result of the model that used 100,000 data when I attempted utilizing the complete dataset. However, the second epoch saw an error since the computing resources were exceeded. Therefore, a better outcome can be obtained if more data are employed.
Secondly, each data has a subcategory named 'subreddit', and the data of the same subcategory explain topics that may be connected. The 100,000 pieces of data I chose were chosen at random, and if they belong to one or two 'subreddit', the accuracy might improve, as it adds context knowledge to the training process.
Thirdly, the comment is a reply to the parent comment so they are related. If the parent comment is included during training which means more context information is added, the model can understand the comments better.
Fourthly, there are some emojis and unusual linguistic features such as caps, italics, or elongated words in the data. For example, "Yeahhh, I'm sure THAT is the right answer", and "**wow**" stands for the bold "wow", the usage of certain features and emojis gives strong indication of sarcasm. But during the pre-processing, they could be restored to their original condition or eliminated. So if the pre-processing techniques are improved, the model's performance could be better.
Fifthly, during the experiment, I simply adopted 0.5 as the threshold value of classification, which may lead to the result being not optimal. Therefore, I should use some methods such as ROC Curves and precision-recall Curves to find the optimal threshold.

## 6 Conclusion

In this study, I tested the sarcasm recognition capabilities of CNN, LSTM, and Bert models. To aid in the detection of sarcasm, I additionally trained three distinct models for extracting sentiment, mood, and personality traits. The results demonstrate that Bert was the most effective in this detection and that the addition of sentiment, emotion, and personality traits did not enhance Bert's performance. Most likely because the three models' ability is insufficient to offer information. Therefore, the late work could be to use better models to extract these three features. And as discussed in 5, employing more data, adding more context information during training, improving the pre-training methods and finding optimal threshold could also contribute to get a better result in the future.

## 7 Links to external resources

Sarcasm Dataset: https://www.kaggle.com/datasets/danofer/sarcasm
Sentiment Dataset: https://www.kaggle.com/datasets/saurabhshahane/twitter-sentiment-dataset
Emotion Dataset: https://www.kaggle.com/datasets/praveengovi/emotions-dataset-for-nlp
Personality Dataset: http://web.archive.org/web/20160519045708/http://mypersonality.org/wiki/doku.php?id=wcpr13

Bert preprocessor: https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3
Bert model: https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4

## References

Corinna Cortes and Vladimir Naumovich Vapnik. 2004. Support-vector networks. *Machine Learning*, 20:273–297.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Mikhail Khodak, Nikunj Saunshi, and Kiran Vodrahalli. 2017. A large self-annotated corpus for sarcasm. *arXiv preprint arXiv:1704.05579*.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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.

Yash Mehta, Navonil Majumder, Alexander Gelbukh, and Erik Cambria. 2020. Recent trends in deep learning based personality detection. *Artificial Intelligence Review*, 53(4):2313–2339.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2016. A deeper look into sarcastic tweets using deep convolutional neural networks.

J. Ross Quinlan. 1986. Induction of decision trees. *Machine learning*, 1(1):81–106.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 704–714.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.