Detecting sarcasm in online forum text using DistilBERT

Jason Polychronopoulos
Student ID: 921565

Jiarui Chen
Student ID: 988280

Benjamin Phommasone
Student ID: 912438

Abstract

Sarcasm detection is a specialised form of sentiment analysis, one that presents challenges in detection even among humans due to sarcasm conveying meaning implicitly. This report explores and highlights the utility of pre-trained models such as BERT (5), training on a distilled version of the full BERT model (11) to be used for classification. This report also extracts hidden layer embeddings from this distilled model in order to evaluate their usage as features in standard classification models. These methods are evaluated over a large corpus of comments collected from the online forum Reddit (6), with instances labelled as sarcastic by comment authors.

1 Introduction

Traditional sentiment analysis typically focuses on determining whether a given piece of text is classified as having either positive or negative sentiment. While this is useful for organisations looking to gain insight into how their product is being received by a wide audience, this form of sentiment analysis may struggle to deal with the sarcastic comments, as the system would not be able to distinguish between literal and implied meaning. As text formats can carry only so much information, the lack of tone and context presents problems in a trained model trying to understand the intent left behind in each online comment.

Previous studies have investigated the role of conversational context in detecting sarcasm (1) (2) (3), such as examining the parent comment which warranted a sarcastic response. Other papers such as Khatri et al. 2020 (4) focused on responses only while extracting embeddings from hidden layers in neural network models that can be used on standard classifiers. The focus of this report is to explore sarcasm detection using the sarcastic comments themselves.

2 Data

The data used is SARC, the Self-Annotated Reddit Corpus (8), which was published in 2017. Reddit is an online internet forum containing public boards for a wide variety of topics and interests which attracts over 400 million monthly users (19). Due to comments lacking tone needed to portray sarcasm, many users opt to self-label their comments with '\s' to denote a sarcastic comment. This data set therefore collates these instances, removing the '\s' characters and labels these comments are sarcastic.

SARC is a corpus of 534 million, of which 1.34 million are sarcastic. This data was scraped between January 2009 and April 2017. The use of this imbalanced data set would make the final model lack critical information about the actual sarcastic comments. Thus, a smaller subset of balanced comments is used instead, which is around one million instances with an equal 50-50 split of sarcastic and non-sarcastic comments. Data between 2009 and 2016 as there were no comments provided for

2017. The data set contains 10 features, each one providing different context about the respective comment. This report focuses on just the sarcastic comments themselves and the class label, as the other features were not relevant to this report's aim.

2.1 Processing

Several data processing steps were taken with the data in order to have it ready for analysis. These involved:

- Removing instances with missing values. These were characterised as empty string values.
- Removing accented letters (eg: 'é') and non-printable characters (eg: '\n'). This was to have all characters as standard ASCII, and non-printable characters would not be able to be analysed properly.
- Removing punctuation. Punctuation may interfere with actual words once tokenised while they do not provide any useful information.
- Convert numeric values to words (eg: '1' becomes 'one'). This was in order to maintain
- Removing capital letters (except words that are all in capitals). This was to have all words consistent while keeping all capital words intact as they might be indicative of sarcasm.
- Removing single letter words (eg: 'a'). These are unlikely to give any useful information.

2.2 Analysis

On first inspection the rate of sarcasm on Reddit seems to drop over time. In figure 1, the rate of sarcasm drops slightly from 58% to 48% between 2009 and 2016. However looking a bit closer it is actually the result of an increase in Reddit usage over time. There is a large increase in the number of comments made each year. In 2016 there were a total of 466,428 comments, which was 47% of the total number of instance in the balanced data set and 256 times more than the number of comments made in 2009.

There is also a behavioural aspect in authors posting in different subreddits. Figure 2 shows the number of sarcastic and non-sarcatic comments in the top five subreddits, and it is obvious that in some of the subreddits have a larger amount of sarcasm than others. For example, "politics" and "worldnews" have more sarcastic comments than non-sarcastic comments. This can make sense, as topics relating to politics and global news stories can be quite divisive which may attract certain comments that are sarcastic. In contrast, comments in "AskReddit" are dominated by non-sarcastic comments. As AskReddit is mainly a platform for posters to ask and answer questions regardless of topic, there may be more serious answers than sarcastic. AskReddit is also the largest subsample of the data set, and as such may just be subject to sampling bias from collection of the original data set.

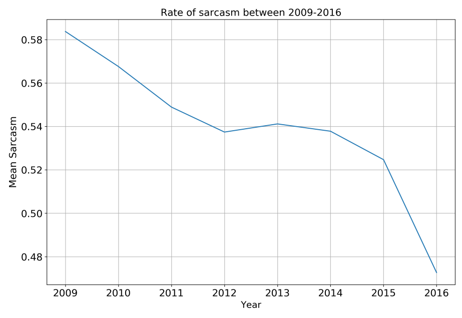

Figure 1: Rate of sarcastic to non-sarcastic comments between 2009 and 2016

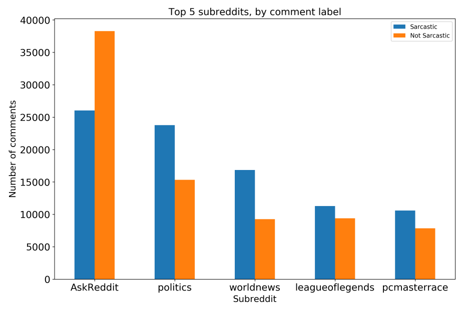

Figure 2: Top five subreddits by total count, grouped by subreddit and label

3 Methodology

After the preparation of the data set, the next step was to investigate the potential techniques and models that could help solve the task. These included using pre-trained language models and bag-of-words approaches.

3.1 Transformers

Introduced in the 2017 paper "Attention is all you need" (10), the transformer neural network architecture aimed at improving the shortcomings of Long Short Term Memory models, or LSTMs. LSTMs were slow to train and relied on model inputs being given in sequential order. For a given sentence, this meant each word had to be added to the model one at a time.

Transformers were able to solve this by defining a model that allows for parallelisation of inputs, where entire sentences may be processed at once. These neural networks utilise an encoder-decoder block architecture. The encoder block is responsible for learning the properties of language and context, while the decoder block is able to take the encoded inputs and map them according to the outputs. Take the task of translating English to German. English words are fed into the encoder block, which are encoded and embedded, while German words are given to the decoder during training. While the encoder block learns about the English language from the inputs, the decoder block transforms the German words and maps them to the vectors output from the encoder block.

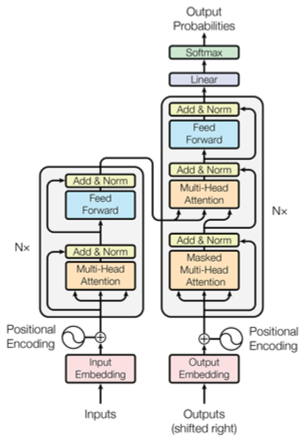

Figure 3: Transformer model architecture (10). Left side is the encoder block, right side is the decoder.

The primary feature of transformers is self-attention, the process of understanding how important a given word in a sentence is to every other word. This is discussed in more detail in section 3.4 on embeddings, though it is enough to understand that by computing the attention vector of each word in a sentence, it enables the capturing of context in language as embeddings for words are determined in relation to surrounding words.

3.2 BERT

Proposed in 2018 by Google researchers (5), BERT is a pre-trained language model that aims to apply the architecture of transformers to language modelling. Essentially BERT is a stacking of multiple transformer encoder blocks, which enables it to gain an understanding of language itself through two training tasks. By pre-training the model on data such as unpublished books and English Wikipedia (14), BERT is able to generalise well across multiple tasks such as sequence classification, of which is the focus of this report.

As noted in the Google paper, standard conditional bidirectional models produce the undesired property of being able to indirectly predict the a target word by looking backwards, essentially "seeing itself". Masked language modelling solves this by randomly masking 15% of the input tokens, and then tries to predict these masked tokens. Next sentence prediction is another training task that gives BERT more knowledge of language. During pre-training BERT takes two sentences, A and B, where one may be in response to another and determines whether sentence B follows sentence A. For instance if BERT was to be pre-trained using Reddit comments, sentence B would be sarcastic remark while sentence A would be the comment prompting the sarcastic remark. By training these two tasks simultaneously BERT is able to attain a good understanding of language, which is then leveraged during fine tuning.

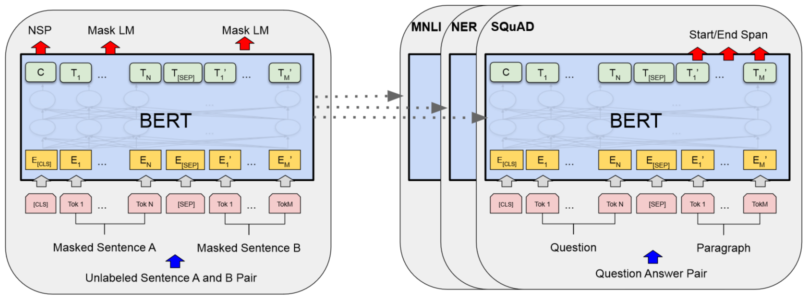

Figure 4: Procedure of pre-training and fine tuning BERT (5) on two masked sentences. Both pre-training and fine tuning share the same architecture.

Fine tuning involves modifying what token sequences are input and the output layers for the desired task. In the diagram above, BERT is fine tuned to answer questions. Both question and answer are input during training, though only the answer tokens $T'_1 \ldots T'_M$ are output. In this report, the output layer is fine tuned to return a prediction of whether an input comment is sarcastic or not, given only a sarcastic response.

3.2.1 DistilBERT

While BERT is very accurate, it is large and computationally expensive. Using a technique called knowledge distillation for neural networks (12) (13), "student" models learn the full output distribution of the full "teacher" model. Thus these students effectively are able to learn the behaviour of the full model. DistilBERT (11) is able to capture almost all the performance of the base model (97% performance retained) while being smaller and more computationally efficient, having 40% fewer parameters and performing 60% faster than BERT.

The usage of a distilled model is both for convenience and the ability to train on more instances in a shorter amount of time. It enables more training runs to be performed, with the decreased performance using a distilled model being made up by training over a larger amount of data in a shorter amount of time.

There are several versions of BERT which have been distilled. **BERT$_{\textbf{BASE}}$** is a model with 110M parameters, 12 attention layers and hidden layer sizes of 768. By contrast, DistilBERT contains 66M parameters and has half as many layers. **DistilBERT$_{\textbf{BASE}}$** is chosen as the primary model for this report.

3.3 WordPiece

BERT employs a tokeniser called WordPiece, a technique introduced in 2012 by Google researchers for Japanese and Korean voice search systems (9). WordPiece pairs tokens together based on how much information will be gained if said pairing was made, where the likelihood is found by the frequency occurrences of other grouped tokens. This approach allows the tokeniser to create sub-words that are shared between multiple different words, for example, splitting the words "pushed", "pushes" and "pushing" into the tokens "push" "##ed", "push" "##es" and "push" "##ing". As the sub-word "push" is shared between all three words, WordPiece will factor this into these groups as this allows all three sub-words to share the same embedding number.

Essentially, this allows WordPiece embeddings to generalise to words that have not been seen in the vocabulary, which enables greater predictive power. In the original paper, BERT had a 30,000 English word vocabulary tokenised using WordPiece.

3.4 Embedding

Embedding is the method of mapping character sequences into numeric values. The benefits of using a tokeniser before this step is that more tokens will now have the same embedded value. Taking in the previous example of the three versions of "push", if these weren't tokenised before, the embedded values assigned to each word would be different despite the fact that they all contain the same sub-word, thus reducing the training effectiveness of these words. But since proper tokenisation steps has been taken already, the sub-word "push" in each of the words will be assigned the same embedded value.

The representation of these values is in the form of an attention vector, where this vector captures an entire sentence and how each value relates to the other words in said sentence.

$$\begin{matrix} The & \rightarrow & The & Quick & Brown & Fox \\ Quick & \rightarrow & The & Quick & Brown & Fox \\ Brown & \rightarrow & The & Quick & Brown & Fox \\ Fox & \rightarrow & The & Quick & Brown & Fox \end{matrix} \Longrightarrow \begin{matrix} [0.72 & 0.05 & 0.08 & 0.17] \\ [0.05 & 0.65 & 0.01 & 0.29] \\ [0.04 & 0.05 & 0.58 & 0.33] \\ [0.04 & 0.05 & 0.04 & 0.87] \end{matrix}$$

This attention vector is able to encapsulate the context of each sentence, as well as the positional embedding of each word. Attention vectors aren't exclusive to transformers, as it's possible to extract these from hidden layers and use to train different models.

3.5 Baselines

Several baseline models were utilised to compare and highlight the strength of DistilBERT as a sequence classification model, both using tf-idf features and embeddings extracted from the last hidden layer of DistilBERT.

3.5.1 tf-idf

One of the feature embedding techniques used to contrast against embeddings extracted from DistilBERT is tf-idf. Tf-idf is a numerical statistic and reflects how important a word is to the collection of the text. Tf-idf consists of two independent statistics: term frequency and inverse document frequency. Tf-idf is defined as

$$\textbf{tfidf}_{i,j} = \textbf{tf}_{i,j} \times \textbf{idf}_i$$

As the literal meaning, term frequency is the frequency of the certain word. For the certain word t_i, the tf_i is calculated by:

$$\textbf{tf}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

where n_i is the count of word t_i in document d_j. The denominator is the total word count in document d_j.

Idf is the inverse document frequency, which is defined as

$$\textbf{idf}_i = \log_{10} \frac{|D|}{|\{j : t_i \in d_j\}|},$$

where $|D|$ is the number of documents and $|\{j : t_i \in d_j\}|$ is the number of the documents which contain the word t_i.

3.5.2 Naive Bayes

Naive Bayes classifiers are a family of probabilistic classifiers based on Bayes' theorem, where For each instance, the class minimises the value of the likelihood function to decide the final classification, i.e.:

$$y = argmax_y P(y) \prod_{i=1}^{n} P(x_i|y)$$

To create a baseline model for DistilBERT, the multinomial event model of Naive Bayes was chosen as this model counts the frequency of words in each comment to determine the class label. This however is not applicable to DistilBERT embeddings as these embeddings contain negative values, which Multinomial Naive Bayes cannot process.

3.5.3 Logistic Regression

Logistic Regression is a generalised linear model that assumes a logit relationship between predictor and response variables. For a binary case with classes c_1 and c_2, this relationship defined as

$$logit(p) = ln(\frac{p}{1-p}) = \beta_0 + \beta_i x_i$$

where p is the probability of choosing value Y such that $p = P(Y = c_1)$.

This model has been shown to be effective at using embeddings from BERT models (4) and as such was used to compare results between embeddings extracted from the layer hidden layer of the BERT neural network and tf-idf features generated from the documents.

3.5.4 Random Forest Classifier

Random forests are an ensemble of decision tree models, where each decision tree is trained on a sub sample of the full training set. Each trees prediction is then voted on, with this reports implementation using soft voting across all trees. A single decision tree may be prone to overfitting, while an ensemble that averages performance over multiple trees results in improved performance and control of overfitting.

4 Experimentation

The experiments were run on local devices, with Nvidia GPUs used to train the neural network models, as cloud services such as Microsoft Azure at the time did not offer GPU instances with their trial versions. The baseline models were evaluated over 808,618 instances, with a 80-20 train-test split. Different train-test splits for DistilBERT were evaluated though no clear pattern was found, thus a split of 90-10 was used due to it’s high reported accuracy.

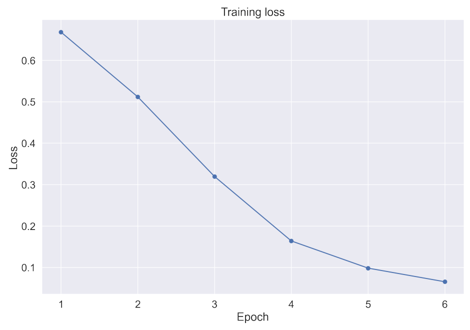

Figure 5: Training loss over a 60-40 split

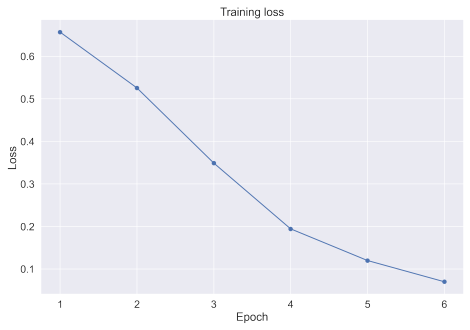

Figure 6: Training loss over a 90-10 split

As it was computationally expensive to run DistilBERT for a large amount of instances over multiple epochs, training loss over multiple epochs was investigated using a trivial amount of instances. During development, it was found that generally between two to six epochs produced good enough validation performance in training. Training loss would sharply decrease until around the 4th, where from there it would begin to flatten out.

Extracting embeddings from DistilBERT involved simply training a DistilBERT model for one epoch, saving the embeddings generated for each batch of instances that passed through the model. These embeddings are taken from the last hidden layer in the DistilBERT model, before they are transformed via an activation function into a probability distribution. This method is similar to (4), who use BERT as a service to generate embeddings via the ‘bert-as-service’ (18) package. The embeddings for 224,944 instances were generated and saved to be used in the baseline models.

4.1 Hyper-parameters

Hyper-parameters for baseline models were determined by performing grid search over a hyper-parameter space. The chosen parameters are detailed in the table below. Of note, random forests tended to choose the largest possible values for their parameter spaces. This would indicate that the random forest would always want to have the deepest and most amount of trees possible, though this could also be due to the search space. Logistic Regression models favoured stronger regularisation, which would indicate a tendency for better generalisation in the models.

Table 1: Hyper-parameters for baseline models

Feature method	**Model**	**Parameter**	**Value**	**Search space**
td-idf	LR	C	0.7	[0.5, 0.6, ... 1.5]
	MNB	alpha	6	[1, 2, ... 6, 7]
	RF	max_depth	200	[10,20,50,100,200]
		n_estimators	1000	[100,200,500,1000]
DistilBERT embeddings	LR	C	0.001	[0.001, 0.01, 0.1, 1, 10, 100, 1000]
	RF	max_depth	200	[10,20,50,100,200]
		n_estimators	1000	[100,200,500,1000]

On the other hand, DistilBERT had its parameters tuned qualitatively due to the computation required to evaluate over a hyper-parameter space across large amounts of instances and epochs. As a result, some parameters were tuned to the recommendations of the authors of BERT, such as the learning rate and number of epochs. DistilBERT uses the Adam optimisation algorithm (17) in order minimise loss, using $\beta_0, \beta_1 = 0.9, 0.999$ and learning rate set to 0.00002 after some test runs. Weight decay was implemented for non-bias terms in the neural network as a means to control complexity in the loss function, making it less likely to overfit while keeping biases fixed, and this was set to the default value of 0.01.

4.2 Results

The performance of the classification methods are evaluated using precision, recall and F1-score. Precision is a measure of the proportion of predicted positives that were indeed correct, recall measures the proportion of true positives that were identified correctly and F1 score is the harmonic mean of both. These are defined as

$$Precision = \frac{TP}{TP + FP},\ Recall = \frac{TP}{TP + FN},\ F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

Additionally, Matthews correlation coefficient (15) is used to evaluate the DistilBERT models. Matthews correlation coefficient is a measure of the quality of binary classification systems, essentially analogous to Pearson's correlation coefficient. It is a real value in the range $[-1, +1]$ and is defined as

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN}}$$

+1 indicates a prediction was perfectly correct, 0 that it was no better than a simple random prediction and -1 indicates that there is no agreement between prediction and observation. MCC is generally used as the metric to evaluate models over the CoLA (16) dataset.

Table 2: Baseline model results

		Precision		**Recall**		**F1-Score**		**Acc**
Feature method	**Model**	**S**	**NS**	**S**	**NS**	**S**	**NS**	**S+NS**
tf-idf	Logistic Regression	0.740	0.704	0.685	0.757	0.711	0.730	0.721
	Multinomial NB	0.710	0.689	0.678	0.720	0.693	0.704	0.699
	Random Forest	0.747	0.670	0.614	0.791	0.674	0.726	0.702
DistilBERT embeddings	Logistic Regression	0.758	0.715	0.687	0.781	0.721	0.747	0.734
	Multinomial NB	-	-	-	-	-	-	-
	Random Forest	0.757	0.714	0.686	0.781	0.720	0.746	0.734

Table 3: DistilBERT model results

		Precision		Recall		F1-Score		MCC	Accuracy
Train count	**Epochs**	**S**	**NS**	**S**	**NS**	**S**	**NS**	**S+NS**	**S+NS**
100,000	6	0.750	0.754	0.745	0.759	0.748	0.757	0.504	0.752
200,000	6	0.779	0.782	0.784	0.777	0.781	0.780	0.561	0.781
400,000	2	0.772	0.757	0.751	0.778	0.761	0.767	0.529	0.764

5 Discussion

It should be noted that while not displayed in this report, raw text comments performed better on the pre-trained DistilBERT model and embeddings than ones that had been pre-processed using the methods described in 2.1. This could be due to WordPiece and attention vector features providing all the cleaning needed, without any need to remove things like stopwords and single letter words.

On all the applicable baseline models, the ones utilising the extracted DistilBERT embeddings performed better than the ones employing the tf-idf features. While tf-idf is simple to implement and interpret, this bag-of-words approach tends to lack in areas that the DistilBERT focuses on, hence this gap in performance. For example the lack of positional embeddings is most likely the biggest offender as it is expected the more context a model is able to learn, the better it can theoretically perform.

There is an interesting difference between the performance of the DistilBERT models that were trained on 200,000 and 400,000 instances. While it is reasonable to expect that more training data would equal better performance, the model trained over 200,000 instances performed better than the model trained on 400,000. The most important factor separating these two models is that the 200,000 model was trained over 6 epochs, while the 400,000 was only trained for two epochs due to time constraints. However it was found through testing that validation performance did not change over epochs, with the first epoch generally being indicative of the final performance. This would warrant further investigation and testing.

One limitation of the final model is that sarcasm was evaluated only on the sarcastic comments themselves, which ignores the other useful information from the data set. A future consideration would be to look into the context provided by the respective parent comment as well as the forum page said comment was posted on. Several studies referenced in the introduction section already investigated this, so this would have potential. There may also be crucial information or key words located in the parent comment that might trigger a sarcastic response, or a link between certain key words in the parent comment and the forum page the discussion was taking place in.

6 Conclusion

This report has demonstrated the effectiveness of using a pre-trained distilled BERT model, as well as utilising embeddings generated from these pre-trained models over more traditional methods such as using bag-of-words. Embeddings generated from DistilBERT outperformed bag-of-words methods in baselines, while the fine tuned model produced benchmark results with the best F1-score being 0.78, and all MCCs being greater than 0.5 which means .

Recommendations for improving this fine tuned model include taking advantage of more computation resources for training larger models, utilising a wider range of data sources such as Twitter and including contextual information such as parent comments in order to produce a more powerful detection system. BERT has since been improved on by XLNet (20), and as such could be used over BERT based models for this task.

The code for this project can be accessed on Github at `https://github.com/jamuupolyy/Sarcasm-Detection-Transformers`.

References

[1] Ghosh, D., Fabbri, A. R., & Muresan, S. (2017). The role of conversation context for sarcasm detection in online interactions. arXiv preprint arXiv:1707.06226.

[2] Pant, K., & Dadu, T. (2020). Sarcasm Detection using Context Separators in Online Discourse. arXiv preprint arXiv:2006.00850.

[3] Ghaeini, R., Fern, X. Z., & Tadepalli, P. (2018). Attentional multi-reading sarcasm detection. arXiv preprint arXiv:1809.03051.

[4] Khatri, A. (2020). Sarcasm detection in tweets with BERT and GloVe embeddings. arXiv preprint arXiv:2006.11512.

[5] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[6] Reddit, `www.reddit.com`

[7] Ofer, D., Sarcasm on Reddit, `https://www.kaggle.com/danofer/sarcasm`, 2017

[8] Khodak, M., Saunshi, N., & Vodrahalli, K. (2017). A large self-annotated corpus for sarcasm. arXiv preprint arXiv:1704.05579

[9] Schuster, M., & Nakajima, K. (2012, March). Japanese and korean voice search. In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 5149-5152). IEEE.

[10] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[11] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.

[12] Buciluǎ, C., Caruana, R., & Niculescu-Mizil, A. (2006, August). Model compression. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 535-541)

[13] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

[14] English Wikipedia, `https://en.wikipedia.org/wiki/English_Wikipedia`

[15] Matthews, B. W. (1975). "Comparison of the predicted and observed secondary structure of T4 phage lysozyme". Biochimica et Biophysica Acta (BBA) - Protein Structure. 405 (2): 442–451.

[16] Warstadt, A., Singh, A., & Bowman, S. R. (2019). Neural network acceptability judgments. Transactions of the Association for Computational Linguistics, 7, 625-641.

[17] Loshchilov, I., & Hutter, F. (2017). Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.

[18] Xiao, H. (2019). Serving Google BERT in Production using Tensorflow and ZeroMQ. `https://hanxiao.io/2019/01/02/Serving-Google-BERT-in-Production-using-Tensorflow-and-ZeroMQ/`

[19] Reddit Information (2020). `https://www.redditinc.com/`

[20] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In Advances in neural information processing systems (pp. 5753-5763).