Analysis of COVID-Twitter-BERT

Aman Jaiman
Paaras Bhandari

1 Abstract

In our project, we replicate and perform extension studies on the COVID-Twitter-BERT model (CT-BERT). CT-BERT is a pre-trained BERT model that uses COVID-19 tweets as its dataset. We start with a discussion of the technologies that we used throughout the project. Next, we replicate the results of the CT-BERT paper on one of the datasets presented in the research. We then perform two extension studies: hyperparameter tuning and testing with other datasets. We note that the CT-BERT paper reports better improvement on COVID-19 related datasets, as compared to general or Twitter datasets, but we argue that their metric isn't the best to judge improvement. With our experiments, we reveal two insights about the CT-BERT model. Firstly, the model does not always outperform BERT on datasets that are similar to CT-BERT's training data. Secondly, unlike what is reported by the original paper, CT-BERT performs similarly on Twitter data as COVID-19 data. We believe this is due CT-BERT learning general tweet language (abbreviations, slang, etc.) and not just COVID-19 language as a result of it's training data.

2 Introduction

Due to the openness of this project, we decided to approach it in the following way. Firstly, we wanted to explore and learn about transformers and language models to set a solid basis for the rest of the project. Next, we wanted to recreate the results of a language model ourselves - in our case this ended up being a specialized model built on top of BERT. Lastly, we would perform extension experiments on the model to unlock new insights. In our case, we show that the results of the paper do not always hold.

2.1 Transformers

The first thing we explored were sequence-to-sequence models (seq2seq) and transformer architectures. With the rise of neural natural language processing (NLP), more applications utilized recurrent neural networks (RNNs) and long-short term memory networks (LSTMs) (Hochreiter and Schmidhuber 1997; Sutskever, Vinyals, and Le 2014). These models are seq2seq, where the input is a sequence, and general the output is a sequence as well. The architecture consists of encoder blocks, which encodes the input sequence to some representation understood by the model, and decoder blocks, which decode the representation to perform the defined task (such as language modeling which we'll explore in this report).

RNNs and LSTMs both retain past information through hidden states: that is their way of trying to maintain context as it encodes the sequence. The issue with this is that the context of past information is lost quickly, especially notable in longer sequences. For example, in RNNs, the hidden state is only kept for the next time step. Although LSTMs solve this problem by maintain some level of the hidden states at each time step, they still do not maintain context as well as is needed.

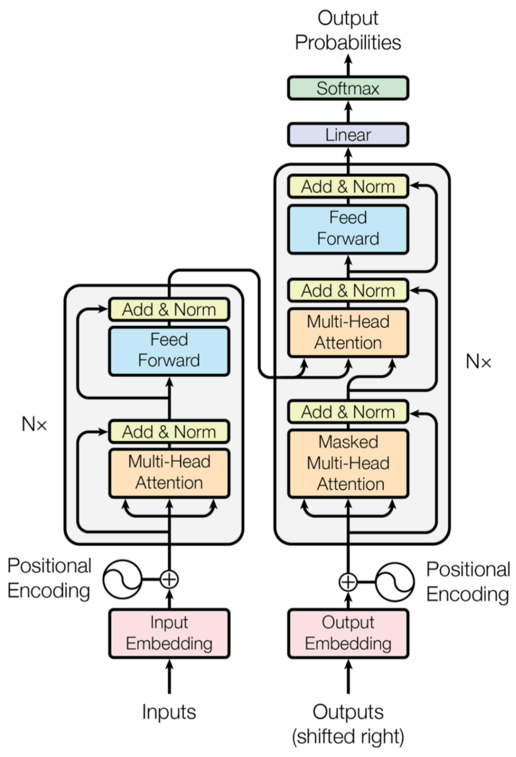

Figure 1: Transformer Architecture (Vaswani et al. 2017)

The transformer architecture (Vaswani et al. 2017) is built upon the same encoder-decoder idea as other seq2seq models. The transformer solves the context problem with the use of self-attention in each of its encoder and decoder blocks. Self-attention is the idea of computing a function that can learn context between words. Using the attention function, the model computes a vector that contains the context rating between different words in the entire sequence. The transformer model then takes this idea further by introducing multiple attention heads. Each head performs its own attention mechanism with independent weights. This way the different heads can learn different representations, resulting in a deeper understanding of language by the model.

The transformer also has positional encodings. While RNNs and LSTMs parse the sequence one word at a time, transformers look at the entire sequence at once. This means they need to encode some information about the position of the words, which are done by these positional encodings. This is important because a word's meaning can change based on where it is in the sentence. An additional benefit to parsing the entire sequence at once is that transformers can parallelize their learning, while models like RNNs and LSTMs cannot.

2.2 Pre-Trained Models and BERT

While transformers are a great architecture for language modelling, they can take a long time to train. In order to make these networks more accessible for people, researchers began pre-training transformers (Qiu et al. 2020). The pre-trained models are then fine-tuned for various downstream tasks. With transformers, you can pre-train either the encoder or the decoder side. The most popular pre-trained decoder model is the Generative Pre-trained Transformer (GPT), which is primarily used for generative tasks. For pre-trained encoder models, we turn to BERT: Bidirectional Encoder Representations from Transformers (Devlin et al. 2018).

BERT was trained on BooksCorpus and English Wikipedia data, for a total of 3.3 billion words. During pre-training, the main task is masked language modeling. In this task, BERT is given a sequence that has some percentage of input tokens masked with a [MASK] token. The model then tries to predict those masked tokens. In order to account for there not being a [MASK] token in fine-tuning data, the authors change up the masking method randomly: either with the [MASK] token (80% of the time), a random token (10%), or an unchanged token (10%). This prevents the model from overfitting to the token and performing poorly on fine-tuning data for downstream tasks. BERT is also trained with a "next sentence prediction" task, specifically for dealing with relationships between sentences. In this task, the model is given a pair of sentences. 50% of the time the second sentence follows the first, and 50% of the time it does not.

The authors trained two various of BERT, based on the number of model parameters: BERT-base and BERT-large. We will be referring to BERT-large as BERT throughout the paper, as all experiements were done on BERT-large. BERT took 4 days to train on 64 TPU chips, which shows the time and technology that is needed to train large transformer models. The fine-tuned tasks can be trained on single GPUs and take a few hours depending on the task. BERT in particular can be fine-tuned for various tasks such as text-classification, sentiment analysis, and more.

2.3 COVID-Twitter BERT

The BERT architecture has been used for specialized tasks in order to get models that perform better in certain domains. Using the same model and training setup, models can be trained on domain specific data. If there is not enough data, models can start out with the trained weights of a prior model (like original BERT), and start training on the specialized data from there.

Our project looks into COVID-Twitter-BERT (CT-BERT), a model developed in 2020 soon after the COVID-19 pandemic hit in the United States (Müller, Salathé, and Kummervold 2020). The model is built upon BERT, and is pre-trained using the same tasks as the original model. CT-BERT uses a corpus of 160 million tweets relating to COVID-19. The authors create the dataset themselves, then train the model on a single TPU for 120 hours.

The paper tests CT-BERT on five downstream classification tasks. Three of the tasks dealt with tweets that had COVID-19 related language: classification of news vs. personal opinion (CC), classification of vaccine sentiment (VC), and classification of stances towards maternal vaccines (MVS). Two other tasks were more general: Twitter Sentiment SemEval (SE) and Stanford Sentiment Treebank 2 (SST-2). They ran all the tasks on both CT-BERT and BERT and compared their performance. They found that CT-BERT outperformed BERT across all tasks.

3 Replication Study

Our first main project task was to do a replication study on the paper to see if their results held on at least the datasets they tested on. Initially we wanted to train the original BERT model on the large COVID twitter dataset, however we were deterred by how long it would take, especially because we did not have the same technology to run as they did. We decided to pull their model from HuggingFace (Wolf et al. 2019), a Python library that provides pre-trained NLP models.

Of the five models that the paper tested on, only three are available publicly as of now: VC, SST-2, and SE. VC and SE resulted in issues while downloading because Twitter's API requires a timeout before API calls, so getting the data would have taken days. We decided to test SST-2, and if time permitted test VC and SE (which we sadly did not get to).

We follow the standard training procedure as in the paper. We train the model with the full SST-2 dataset and a batch size of 8. Training was done for 3 epochs using a learning rate of 2e-5. Our final validation accuracy comes within 0.005 of the paper's recorded results.

4 Extension Study - Hyperparameter Tuning

For both of our extension studies, we use Google Colaboratory (Colab) Pro. This allows us to train multiple models at once, as well as use hosted GPUs rather than use our personal machines. The GPUs that were used would change based on the Colab runtime, but they were either a Tesla T4 or a Tesla P100-PCIE-16GB.

The first part of our extension study dealt with looking at how various hyperparamters could change the performance of the CT-BERT model. We aimed to try to either improve performance or at least find some insights about the hyperparameters relationships in training. The following hyperparameters were tuned: learning rate, percentage of data, batch size, and number of epochs. All experiments in this section were performed using the SST-2 dataset.

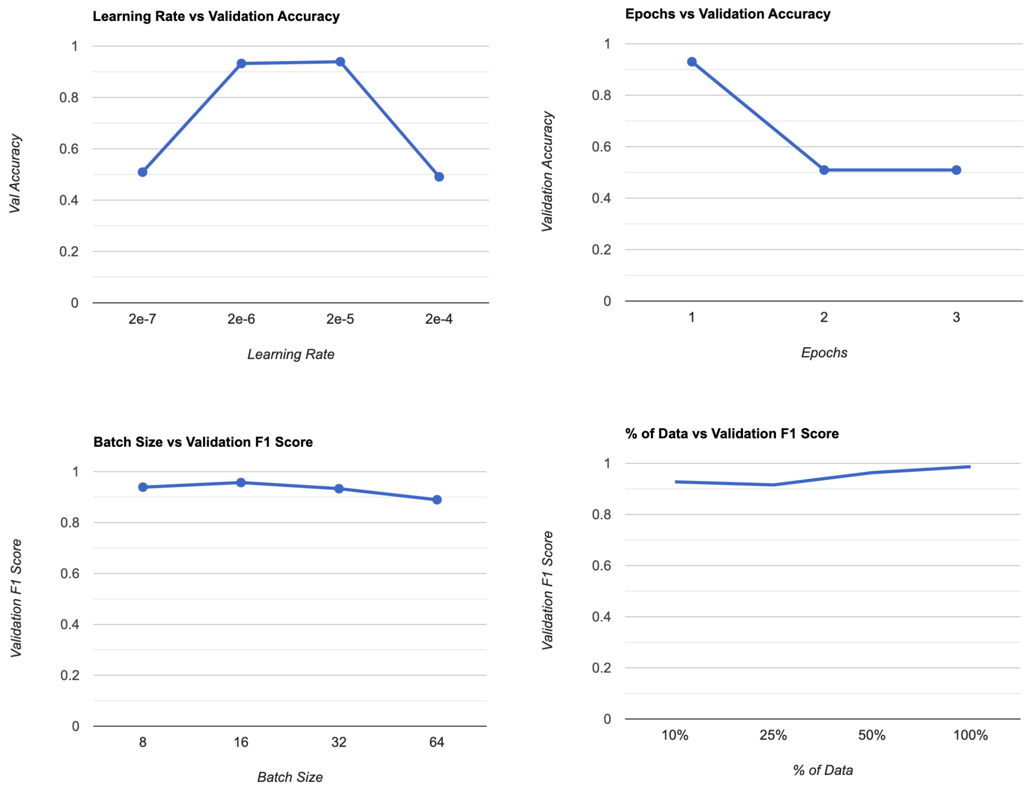

Figure 2: Graphs for each hyperparameter tuning

4.1 Learning Rate

The model's performance peaked at learning rates 2e-5 and 2e-6, but dipped sharply for learning rates outside this range. As mentioned on the official BERT Github, 2e-5 is the optimal learning rate when pre-training models. We do see that at 2e-6, the model outperforms CT-BERT slightly: an F1 score of 0.957 compared to 0.944. Outside of this, we find that the model was quite sensitive to learning rate adjustments. Due to these large accuracy drops, the F1 score returned was nearly 0 for learning rates 2e-4 and 2e-7.

One additional set of experiments we tried here was taking the best learning rate we found, 2e-6, and adjusting another hyperparameter, in particular the batch size. Unfortunately we found that no adjustment to the batch size would increase the performance with the new learning rate.

4.2 Epochs

The next hyperparamter we adjusted was the number of epochs. Initially, when using 100% of the dataset, we were able to get a good performance with a single epoch, after which the model started to overfit and the performance drops sharply. We see the accuracy near 50%, resulting in extremely low F1 score. However, with lesser data, the model did not tend to overfit. When we test with just 1.4k training examples, we see that training for more epochs helps performance. The F1 score goes from 0.929 to 0.967 when training for 1 to 3 epochs. As we increase the training data to 25% of the data, 17k examples, the model begins overfitting again as the number of epochs increases, but not to the same extent as with 100% of the data. Here, we see the best performance at 1 epoch with an F1 score of 0.982, which drops to 0.956 at 3 epochs. We find that the number of fine-tuning examples can be fairly "small" and only 1 epoch is needed to train to achieve the best performance.

4.3 Batch Size

We also tuned the batch size, starting from 4 to 64, in increments of powers of 2. The default batch size was 8, where we get similar performance to the paper's results. A batch size of 4 gets very poor validation accuracy, resulting in a F1 score of nearly 0. We find that the model's performance peaks at a batch size of 16, achieving a validation F1 score of 0.957, compared to 0.944 of the paper's. Increasing the batch size to 32 or 64, however, worsens the model. Increasing the model's batch size also allowed us to decrease the sequence length. In transformer based models, the sequence length is the number of words in a given training example/sentence. The training time also reduced significantly as batch size increases, so we were able to get a comparable performance at batch size 32 while cutting the training time by 75%.

4.4 Percentage of Data Used

Finally, we tried adjusting the percentage of data, i.e. how much of the total available data was used for training. For the SST-2 dataset, the total number of training examples were 67k. This meant that even fine-tuning the model could take almost an hour per epoch. While decreasing the training data size does lower the F1 score, the change is not by much. Training with just 50% of the data, which reduces the training time to half, only reduces the F1 score from 0.987 to 0.963. Going as low as 10% of the data, we still achieve nearly 0.93 validation F1 score. For such fine-tuning tasks, we feel that the slight decrease in performance may be worth the faster training time. For all experiments here, we maintained the same validation set of 500 examples.

5 Extension Study - Exploring Other CT-BERT Results

The next part of our extension study was to verify the results of CT-BERT across other domains. In particular, we decided to run experiments to test two questions:

1. How well does CT-BERT's performance translate to other datasets? Do the results shown in the paper hold across other COVID-19 and Twitter datasets?

2. Does CT-BERT perform better on COVID-19 related language, or Twitter language in general? We believe that because CT-BERT was trained on tweets, it has learned general language in tweets just as well, or better, than COVID-19 language. The original paper measures performance using a ΔMP metric which allows them to say that CT-BERT improves performance more for COVID-19 language than general Twitter language. We believe that this metric may not be the best indicator of performance increase, and therefore the model does learn things such as slang, acronyms, etc. to a similar level.

Dataset	BERT	CT-BERT	% F1 Increase
CC (COVID-19 Twitter)	0.931	0.949	1.93%
VC (COVID-19 Twitter)	0.824	0.869	5.46%
MVS (COVID-19 Twitter)	0.696	0.748	7.47%
SST-2 (General Text)	0.937	0.944	0.75%
SE (General Twitter)	0.620	0.654	5.48%

Table 1: Percentage Increase in F1

5.1 Results Analysis

The authors of the paper define a metric called the "relative improvement in marginal performance." The metric is defined as follows:

$$\Delta MP = \frac{F_{1,\text{ BERT}} - F_{1,\text{ CT-BERT}}}{1 - F_{1,\text{ BERT}}} \tag{1}$$

Based on this metric, they report that the CT-BERT model performance increases more for tweets related to COVID-19 than for general tweets. We find two problems with this metric. Firstly, the numerator shows $F_{1,\text{ BERT}} - F_{1,\text{ CT-BERT}}$. In this case, the ΔMP should be negative as CT-BERT performs better than BERT for every dataset. This is not how the values are reported so there is some inconsistency here. Secondly, based on their formula, as the performance of BERT increases the ΔMP would increase as well since the denominator approaches 0. We argue that this should not be as a given difference in score is less significant the larger the score is. In order words, the same marginal improvement should be greater for smaller BERT scores, not larger scores.

We decide to use "percentage increase" between the F1 scores to measure how much better CT-BERT does for the tasks.

$$\%\text{Increase of F1 Score} = \frac{F_{1,\text{ CT-BERT}} - F_{1,\text{ BERT}}}{F_{1,\text{ BERT}}} \tag{2}$$

Table 1 shows the percentage increase of the F1 scores between the two models. Based on this, CT-BERT's performance is **not** better for COVID-19 related tweets when compared to general tweets. We find that the percentage increase for the SE dataset is higher than the CC and VC dataset.

5.2 Testing on Other Datasets

This part is broken up into three sections: other COVID-19 datasets, other Twitter datasets that do not deal with COVID-19, and a performance analysis to determine how well the model does on the two types of data.

5.2.1 Other COVID-19 Datasets

Our first tests focused on other datasets related to COVID-19, as we wanted to see if the paper's results would hold for other datasets. The first dataset we looked at was "Coronavirus tweets NLP - Text Classification" (Miglani 2020) which was publicly available on Kaggle. This dataset contains tweets related to COVID-19 mostly from March 2020. The tweets are labelled

	Epochs									
	1	2	3	4	5	6	7	8	9	10
CT-BERT	0.985	0.982	0.988	0.985	0.985	0.988	**0.990**	0.987	0.983	0.988
BERT	0.972	0.977	0.982	0.980	**0.984**	0.983	0.978	0.976	0.979	0.973

Table 3: F1 Scores on ANTiVax Data

for sentiment as extremely positive, positive, neutral, negative, or extremely negative. We pre-process the data to remove null values, links, hashtags, user mentions, and emojis. The big difference in this dataset is that there are 5 labels, while the CT-BERT paper used datasets that had 2 or 3 labels (with the exception of MVS that had 4).

We ran both CT-BERT and BERT-Large on the dataset for 10 epochs. In order to do so, we slightly modified the architecture by adding an input layer (which adds attention masks) before the model we use, a dropout layer to reduce overfitting, and an output layer with Softmax. The accuracy results are found in Table 2. We find that CT-BERT actually **does not** outperform BERT on this dataset, which goes against the paper's findings. We would expect that this dataset, which is both related to COVID-19 as well as tweet structure, should be similar to the training data of CT-BERT and therefore CT-BERT's performance should be higher.

Model	Accuracy
CT-BERT	0.857
BERT	**0.892**

Table 2: Accuracy on Coronavirus Tweets Data

Next, we look at the ANTiVax dataset (Hayawi et al. 2022). This dataset contains COVID-19 vaccine-related tweets and labels them as misinformation or not. These tweets were pulled from November 2020 until July 2021, unlike the training data for CT-BERT which contained tweets before May 2020. We perform the same data pre-processing as above, which results in a final training set of 10k training examples and 1k testing examples. We use the same model architecture once again, and run both BERT and CT-BERT versions for 10 epochs.

The results show two things. Firstly, the BERT based model increases performance until epoch 5, after which the model starts to overfit to the training data and the validation F1 score drops. This is unlike the CT-BERT based model, where the performance stays around the same level for the entire run. Secondly, for this dataset CT-BERT perform slightly better than BERT. Even for the best run (the 5th epoch for BERT), the F1 score is beat by nearly every epoch of the CT-BERT model. This is to be taken with a grain of salt, however, as the percentage increase between the two best scores is only 0.6%, which is lower than any the increase on any dataset used in the CT-BERT paper.

Our results on these two datasets, shown in Table 3, show that extending the CT-BERT model is not consistent across datasets related to COVID-19. Even though the purpose of the model is to perform better on COVID-19 data, we find that it may not be the case depending on the dataset.

5.2.2 Twitter Datasets

Next, we shifted our focus to other Twitter datasets. Our primary goal here is to see how the CT-BERT model performs when it sees Twitter data that is not related to COVID-19. This allows us to check its perform in a different domain, and also compare results to the original paper's.

The first dataset we look at is the Twitter Sentiment Analysis dataset (Hussein 2021). This dataset is publicly available on Kaggle. The dataset has three sentiments: positive, neutral, and negative. It contains no COVID-19 language, and is mostly related to Indian news. The dataset has 123k training examples and 15k testing examples.

We also look at the Twitter data from A n.d. This data contains tweets in context of the Indian General Elections in 2019. The tweets are annotated as positive, neutral, or negative, and we extract 25k samples to speed up the training time. The dataset is pre-processsed in the same way as the other Twitter datasets.

For both datasets, we find that CT-BERT performs slightly worse than BERT-Large. This comes as a little bit of a surprise to us since we expect CT-BERT, which has learned tweet language and structure, to perform better than BERT, which was never trained on short, slang-ish phrases. The performance is nearly comparable, but sadly it does show two other datasets where CT-BERT's original results do not hold.

Model	Accuracy (Twitter Sentiment)	Accuracy (Indian General Elections)
CT-BERT	0.968	0.869
BERT	**0.975**	**0.885**

Table 4: Accuracy on Twitter Sentiment Analysis Data and on Indian General Elections Tweets

5.2.3 Performance Comparison

As mentioned before, the CT-BERT paper shows results that make it seem that CT-BERT increases performance more for COVID-19 language compared to other datasets. We have already shown why we don't believe the ΔMP metric is a fair metric, and when we go off of the % increase, we find that the conclusion does not hold.

We can also see how CT-BERT performs on the two types of datasets. For example, we compare the performance on the Coronavirus Tweets dataset to the performance on both non-COVID-19 Twitter datasets. While CT-BERT doesn't outperform BERT on either of these datasets, we look at how well the model does relative to BERT, specifically how much accuracy is maintained using CT-BERT instead of BERT. The results of this are shown in 5 We see that CT-BERT does better relative to BERT for both of the other Twitter datasets. Based on this, we conclude that CT-BERT learns Twitter language at least as well as it learns COVID-19 language. It could therefore be used for any Twitter task, not just COVID-19 related tasks, to achieve similar performance as BERT.

Dataset	% of Accuracy Retained
Coronavirus Tweets	96.1%
Twitter Sentiment Analysis	**99.3%**
Indian General Election Tweets	**98.2&**

Table 5: How CT-BERT Does Relative to BERT

6 Conclusion

Through our project, we were able to develop a foundation for further research in language models. Our recreation of the COVID-Twitter-BERT model proves useful in understanding pre-trained BERT-based models. We were able to provide insights into the extensiblity of this particular model. Sadly, we see that the performance increase over BERT is not maintained across various datasets, both related and not related to COVID-19. We also argue that the performance metric used by the CT-BERT paper is not the best way to measure improvement. Using % increase over BERT's F1 score, we show that the model performs just as well on Twitter datasets as COVID-19 datasets. Since the model is trained on tweets related to COVID-19, we believe that the model does a good job of learning tweet structure/language along with COVID-19 language, resulting in similar performance across the two domains.

7 Future Work

We think there can be more work done in this area to solidify our findings, and maybe extend our project further. Currently, we did most of our testing on sentiment analysis classification tasks. This was because it was easier to find these datasets for COVID-19 and Twitter data. In the future, more work can be done with other types of tasks that BERT and CT-BERT can perform, to see if CT-BERT performs certain tasks better (question answering, sentence prediction, etc.). Additionally, it would be interesting to see how much data in a specific domain is needed to create a specialized BERT model that can outperform BERT in that domain. CT-BERT can outperform BERT for some datasets, but adding more training data (or increasing the training time) could make it more consistent across other datasets in the domain.

References

[1] Chaithanya Kumar A. *Twitter and Reddit Sentimental analysis Dataset.* URL: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification.

[2] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *CoRR* abs/1810.04805 (2018). arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

[3] K. Hayawi et al. “ANTi-Vax: a novel Twitter dataset for COVID-19 vaccine misinformation detection”. In: *Public Health* 203 (2022), pp. 23–30. ISSN: 0033-3506. DOI: https://doi.org/10.1016/j.puhe.2021.11.022. URL: https://www.sciencedirect.com/science/article/pii/S0033350621004534.

[4] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[5] Sherif Hussein. *Twitter Sentiments Dataset.* 2021. DOI: 10.17632/z9zw7nt5h2.1.

[6] Aman Miglani. *Coronavirus tweets NLP - Text Classification.* 2020. URL: https://www.kaggle.com/datasets/datatattle/covid-19-nlp-text-classification.

[7] Martin Müller, Marcel Salathé, and Per E Kummervold. “COVID-Twitter-BERT: A Natural Language Processing Model to Analyse COVID-19 Content on Twitter”. In: *arXiv preprint arXiv:2005.07503* (2020).

[8] Xipeng Qiu et al. “Pre-trained Models for Natural Language Processing: A Survey”. In: *CoRR* abs/2003.08271 (2020). arXiv: 2003.08271. URL: https://arxiv.org/abs/2003.08271.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. “Sequence to Sequence Learning with Neural Networks”. In: *CoRR* abs/1409.3215 (2014). arXiv: 1409.3215. URL: http://arxiv.org/abs/1409.3215.

[10] Ashish Vaswani et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[11] Thomas Wolf et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771. URL: http://arxiv.org/abs/1910.03771.