Multi-class Sentiment Analysis of CoVid-19 Tweets

Silvey Yu ly1164, Yunzhe Sun sy2825

Abstract

How are people feeling in the pandemic of CoVid-19? How can we accurately know this? People's emotions are well reflected on social media. That's why we selected Twitter text as our corpus. **Can we do detailed emotion classification on them?** This problem is meaningful because it transforms massive data into informative sentiment results, providing insights for public health policy makers, politicians, economists, psychologists and other researchers. Due to the large computation and noisy nature of language, we aimed to implement 4 models: **SVM, RNN, LSTM, BERT Transformer**, and test their accuracies. We found that .

Introduction

Tons of comments related to the real-life events are sent in social media. Twitter has a word limit for posting, which allows us to focus more on sentence-level and short-comment level sentiment analysis. Under the context of CoVid-19, we tentatively address the problems: what models can we use to classify twitter text into detailed emotion categories? how accurate can we do it, and how to improve accuracy in the future? The main challenge for us is the data has emotions labeled in **5 specific categories: fear, anger, sadness, joy, and neutral**, which is harder than simple "positive and negative" classification as many models did in the previous researches. To solve this challenge, we used **deep learning** methods, because naive methods can work poorly on such specification. Therefore, it is computationally costly and time-wise challenging to tune hyper-parameters and select best configuration for deep learning models.

Dataset

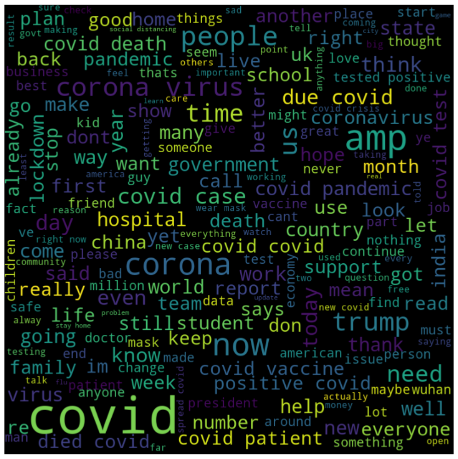
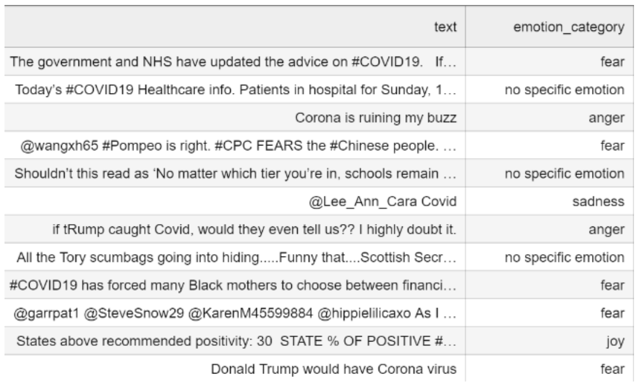


Figure 1: Wordcloud from Overall Data

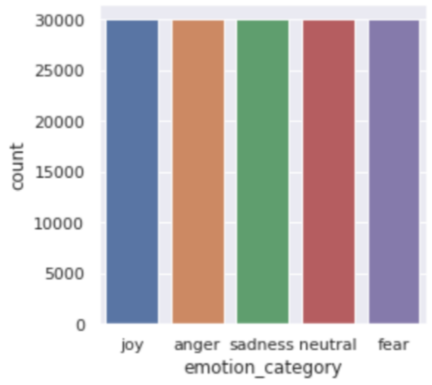
We used the COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes from Open ICPSR. Each tweet is explicitly prelabeled in five specific categories: fear, anger, sadness, joy, and neutral, and the left table in Figure 2 shows a few examples of the original dataset.

Word Cloud provides a significant way to find out the word that often appears in the sentiment reviews. It gives us an initual insight into dataset. Fig.1 is a visualization on the overall dataset. The size of the word represents the frequency of the words appear in the overall dataset.



	text	emotion_category
	The government and NHS have updated the advice on #COVID19. If…	fear
	Today's #COVID19 Healthcare info. Patients in hospital for Sunday, 1…	no specific emotion
	Corona is ruining my buzz	anger
	@wangxh65 #Pompeo is right. #CPC FEARS the #Chinese people. …	fear
	Shouldn't this read as 'No matter which tier you're in, schools remain …	no specific emotion
	@Lee_Ann_Cara Covid	sadness
	if tRump caught Covid, would they even tell us?? I highly doubt it.	anger
	All the Tory scumbags going into hiding.....Funny that....Scottish Secr…	no specific emotion
	#COVID19 has forced many Black mothers to choose between financi…	fear
	@garrpat1 @SteveSnow29 @KarenM45599884 @hippielilicaxo As I …	fear
	States above recommended positivity: 30 STATE % OF POSITIVE #…	joy
	Donald Trump would have Corona virus	fear

(a) Table: Original Tweets and Its Emotion Category



(b) Distribution of Emotion Category

Figure 2: Dataset Features

As for the data, we randomly sampled 30000 tweets from each emotion category. In the sample, we sample 80% as training data, and the rest 20% as test data. In the training data obtained, we further sample 20% as validation data. Notebaly, the same distribution of emotion category is the foundation of correctly building models to classify the emotions.

Besides, we also generated word clouds for each categories. The Fig.3 is an example of word cloud for ”anger”, if you look carefully into it, you will find some typical words related to emotion ”anger”.

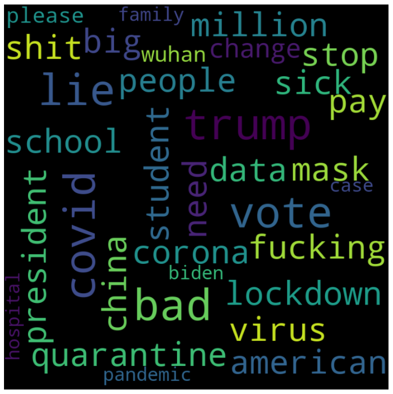


Figure 3: Selected Word Cloud for Emotion Category ”anger”

Solution

1. Proprocessing:

Twitter raw text contains "@", urls, emojis, text emojis, symbols, and improper spellings. We created a proprocessing script to clean data. Besides, commonly used "stop words" can lower model performance. Therefore, to clean them, we utilized 4 python packages (1) nltk.corpus stopwords, (2) pipy (3) SpaCy (4) gensim STOPWORDS. The final cleaned data is separated into train, validation, and test groups, which are used as the input data files in our four models.

Status	Text
Before	I contracted Covid-19 during my 60th birthday celebration – Kennedy Agyapong reveals https://t.co/lxSWuAbJsN via @The N-Connect Media BBNajia
After	i contracted covid during my th birthday celebration – kennedy agyapong reveals via nconnect media

Table 1: An Example of Preprocessing Data

2. Word embeddings:

Global Vectors for Word Embeddings(GloVe) is used in RNN and LSTM to obtain the vector representations for words. Based on the irregularities of the text data structure, the process of text mining requires some initial stages in which the point is to prepare thetext to be more structured. We chose a pre-trained Word2Vec for twitter texts: glove.twitter.27B.100d.txt. Using pre-trained embedding vectors can result in better performance and faster convergence. Each sentence's embedding has a fixed max length and would look like [0, 0, ... ,0, integer$_1, integer_2, ..., integer_k$].
BERT Tokenizer and BERT word embedding are used for BERT Transformer model.

3. Loss function:

Because we are dealing with multi-class classification, for RNN, LSTM and BERT Transformer, we all used categorical cross entropy as our loss function.

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}1_{y_i \in C_c} log[p_{model}(y_i \in C_c)]$$

We double sum over observations i and categories c. $1_{y_i \in C_c}$ is an indicator that observation i belongs to the category c.

4. The four Models we implemented:

Model 1: SVM

SVM performs well in high dimensionality and is widely used for textual polarity detection. Hence, a multiclass SVM model is built to show whether SVM can learn and classify the pattern in the context of sentiment analysis.

Due to the extensively time consumption for the data fitting of SVM model, we conducted Principle Component Analysis to find whether we can use data with small dimensions to represent the most of information needed. However, as the Fig.4 in the right side shows, each of the principal components almost contributes equally to the variance explanation (blue line), which means it is impossible to do dimensionality reduction to reduce the complexity for our word vectors.

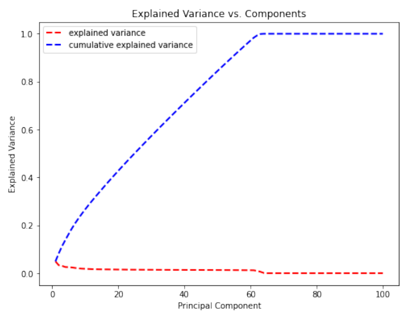


Figure 4: Explained Variance in PCA

Then, we compared different hyperparameters in SVM to optimize the model in terms of the kernel function and regularization parameter C, separately. Based on the classification accuracy in validation data, we selected the best configuration with Gaussian Radial Basis Function (RBF) as kernel function and C=1. The comparison results are as follows:

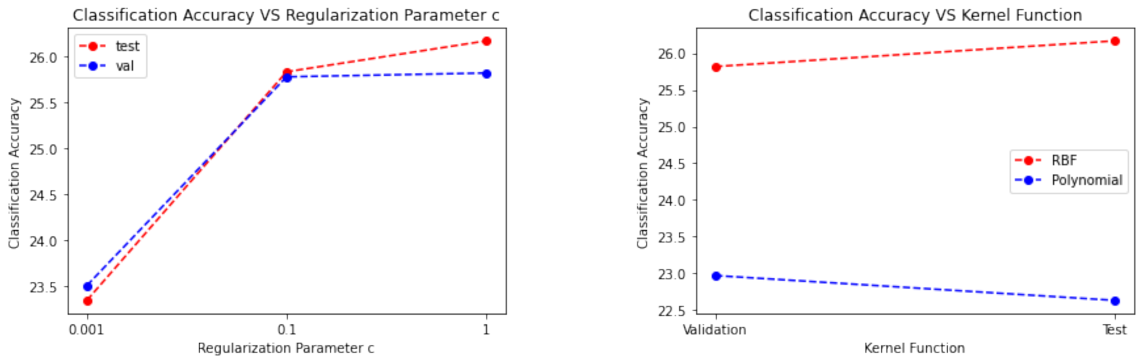


Figure 5: Hyperparameter Tunning for SVM Model

Fig.5 indicates that kernel function plays a significant role in performance. Notably, in the best model, the final classification accuracy of the testing data is 26.67%, which is slightly greater than the random guess among five categories. We concludes that although SVM model may work well in high dimensionality, it does not fit in the sentiment analysis's context. Therefore, we aimed to classify the emotion category with neural network, which can solve the nonlinear problems taking the advantages of multi-layers.

Model 2: RNN

Besides the traditional machine learning model, due to the complex nature of language, we further consider to what extent the neural network can outperform sentiment analysis. To begin with, we used a Keras built-in layer named SimpleRNN to build a fully-connected RNN model. The structure of our RNN model is illustrated in Fig.6.

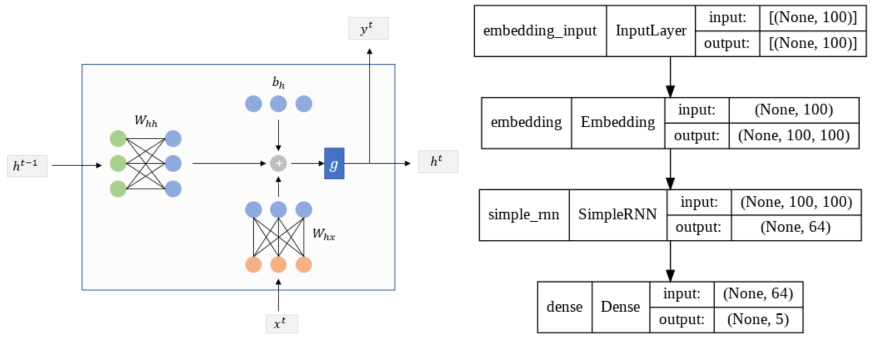

y^t
b_h
W_{hh}
h^{t-1}
g
h^t
W_{hx}
x^t
embedding_input | InputLayer | input: [(None, 100)] | output: [(None, 100)]
embedding | Embedding | input: (None, 100) | output: (None, 100, 100)
simple_rnn | SimpleRNN | input: (None, 100, 100) | output: (None, 64)
dense | Dense | input: (None, 64) | output: (None, 5)

Figure 6: Structure for SVM Model

Then, the hyperparameters: optimizer, epoch, and batch size are tuned separately to find the relatively best model. In the tunning process, the validation accuracy is used as a criterion to choose the optimized parameter. The best configuration is activation function="tanh", optimizer="adam", epoch=16, and batch size=2048. The comparison results are in Fig. 7.

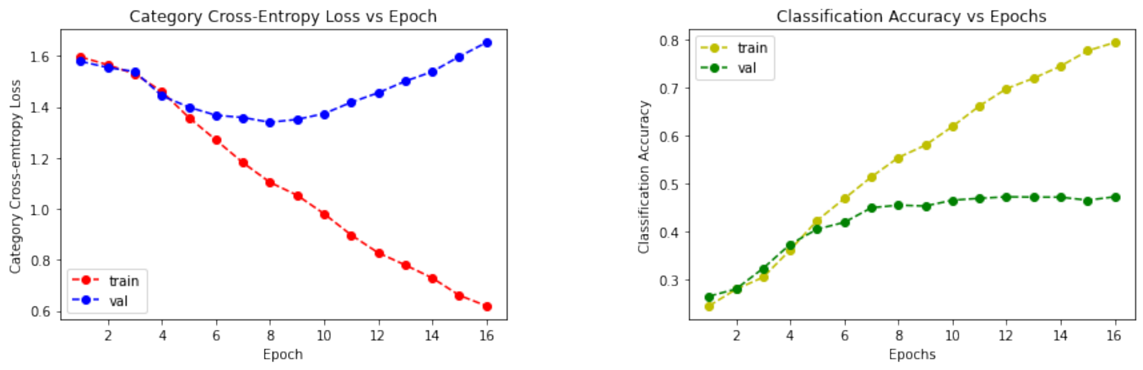


Figure 7: Final Results for SVM Model

To prevent overfitting, in the best model, we applied an early stopping method, where the training process immediately stopped and started to test the model performance if the validation loss is increased. Here, the patience weight is added to prevent the possibility of trapping in local minimum due to the stochasic nature of updating the parameter. Therefore, a callback method is used to reload the best model's parameter. The result shows that the testing accuracy is about

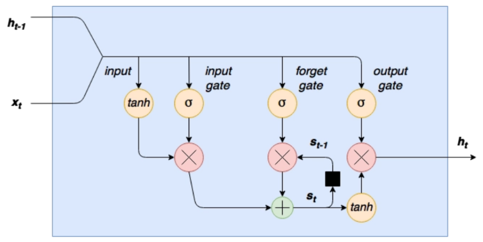
0.43923, twice the value of the random guess, indicating a relatively good model performance for processing sequence data for predictions. However, it may also suffer from its short-term memory due to the vanishing gradient problem. Therefore, to account for this drawback, we next implement LSTM model.

Model 3: LSTM

We used Keras module in Python to build an LSTM recurrent neural network. RNN can suffer from short-term memory, but LSTM efficiently improves performance by **memorizing the relevant information that is important and finds the pattern**, with forget gate and the hidden state updated by it

$$f = \sigma(b^f + x_t U^f + h_{t-1} V^f)$$

$$s_t = s_{t-1} \circ f + g \circ i$$



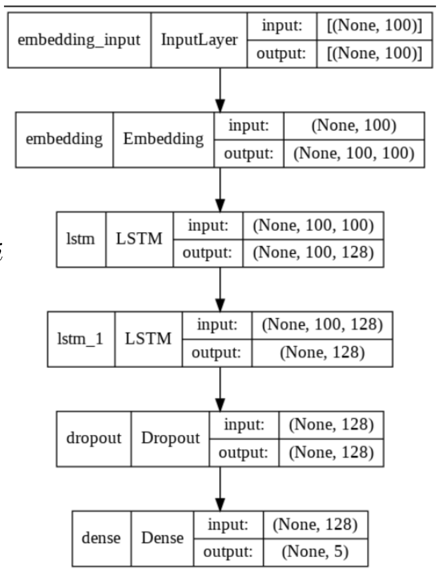

embedding_input | InputLayer | input: [(None, 100)] | output: [(None, 100)]
embedding | Embedding | input: (None, 100) | output: (None, 100, 100)
lstm | LSTM | input: (None, 100, 100) | output: (None, 100, 128)
lstm_1 | LSTM | input: (None, 100, 128) | output: (None, 128)
dropout | Dropout | input: (None, 128) | output: (None, 128)
dense | Dense | input: (None, 128) | output: (None, 5)

Figure 8: Left: LSTM structure; Right: Our own implementation

We did hyperparameter tuning for LSTM. The best configuration is batch size = 1024, activation function = tanh, hidden layer size = 64, and epoch 11 early stopping. The loss and accuracy results are as follows in Fig.9:

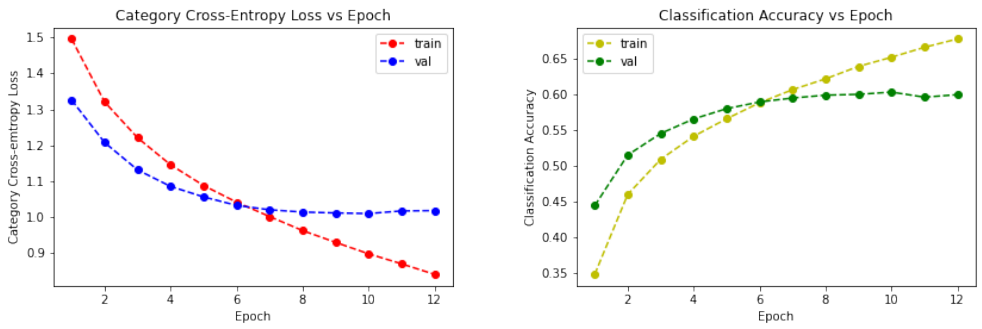
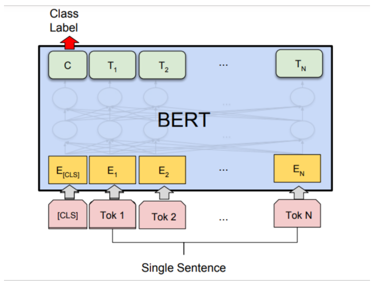


Figure 9: Final Results for LSTM Model

Model 4: BERT Transformer



BERT stands for "Bidirectional Encoder Representation from Transformer". It presented state-of-the-art results It can be applied for a wide variety of tasks such as Question Answering and Sentence Classification can produce high precision. Before feeding word sequences into BERT, a 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence, and thus will be able to learn the context. he Transformer reads entire sequences of tokens at once. While LSTMs read sequentially (left-to-right or right-to-left), the Transformer model is non-directional. The attention mechanism allows for learning contextual relations between words

Thanks to BERT which allows Transfer Learning, short-context tasks such as ours will have a better result using BERT. Even with small amounts of data, the result is rlatively great. In our experiment, BERT starts with a higher validation accuracy than before, and will converge after less than 5 epochs. We early stop the training. The data are shown in the discussions.

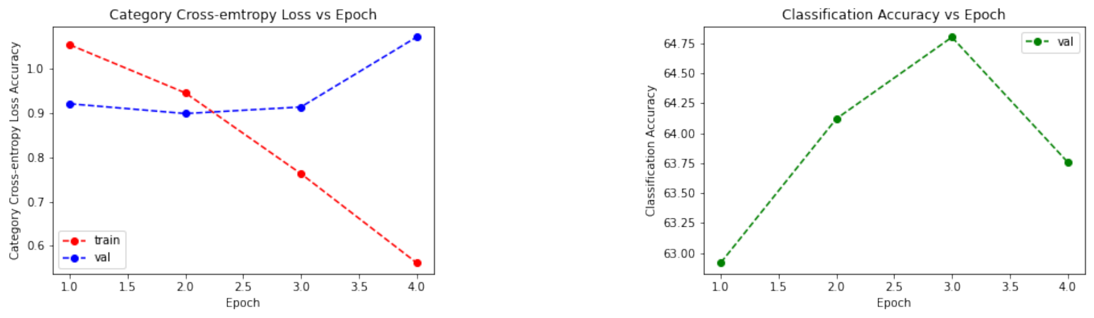


Figure 10
Final Results for BERT Model

BERT is actually a powerful model, despite the difficulty of such detailed classification and the generally fragmented and short tweet texts, it can still achieve high performance in "fear" and "joy" – the 2 most characteristic emotion categories:

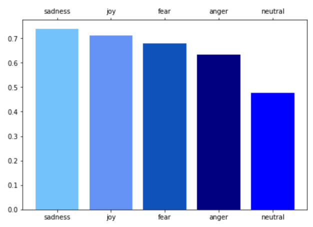


Figure 11: Test Performance per Category

Results and Discussion

The ultimate test accuracy is as follows in Fig.12:

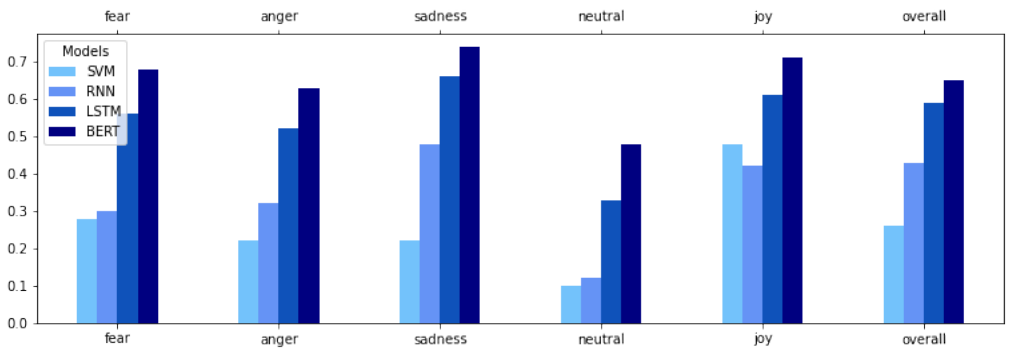


Figure 12: Classification Accuracy per Emotion Category in Different Models

model	fear	anger	sadness	neutral	joy	overall
SVM	0.2832	0.2235	0.2247	0.0982	0.4836	0.2617
RNN	0.2969	0.3157	0.4754	0.1203	0.4164	0.4272
LSTM	0.5597	0.5168	0.6613	0.3275	0.6094	0.5882
BERT	0.6791	0.6327	0.7396	0.4777	0.7122	0.6680

Table 2: Classification Test Accuracy Results

Overall, the performance on test data is: BERT Transformer (66.80%) $>$ LSTM(58.82%) $>$ RNN(42.72%) $>$ SVM(26.167%). As the sixth group of bar graphs indicates, we can observe a gradual performance improvement in these four models, which is consistent with the general impression. Looking deep into each category, generally, the 4 models are poor at classifying neutral tweets.

- **SVM** as a traditional supervised machine learning algorithm performs worst, compared with the random guess among five categories(20%). Specifically, in PCA, the almost equal values of explained variances for different principal components are reasonable. Because the vector representations of 100-dimensions showcase interesting linear substructures of the word vector space. It could be one reason that we could not use a few features to represent most of the information in input data (i.e. a sentence).
- **RNN and LSTM** as neural networks, significantly increase the classification accuracy. From the test and validation accuracies, we can see that LSTM performs better than RNN by 2. When dealing with real-life text pieces, the sequence of data plays an important role. It is beneficial that the model keeps track of relevant information and drops irrelevant ones.
- **BERT Transformer** is designed to support machines understanding the meaning of short context language. Therefore, the performance of BERT transformers is relatively great, and is

the best in this paper. BERT Transformer can achieve convergence after few epochs, and this is an advantage. BERT is also good at predicting the whole context. However, our dataset has too fragmented texts and too common misspellings, and might not be rich enough in emotion features, to develop a highly accurate model, including a BERT model.

Future Improvements

- If we had more time, we could request another twitter dataset SenWave Dataset, which is labeled and checked by 30000 experts, with neat and more complete sentences and more accurate emotion categories. Its access requires author approval, and our time had been too limited to go through the whole process.

- This time, the BERT Transformer model performed the best, and in future works, short-context tasks can also be nicely solved with BERT. However, we did not have enough time to fine-tune the model, because its computation requires GPU acceleration, and even with acceleration, it is slow and memory-intensive. We could run it on an HPC in the future.

- There exists large amounts of common words among the categories, and not enough distinction between them. In future works, preprocessing step could also involve removing some of the most common words.

References

[1] COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions Attributes
[2] Kaggle GloVe txt data
[3] Devlin, J., Chang, M.-W., Lee, K., amp; Toutanova, K. (2019, May 24). Bert: Pre-training of deep bidirectional Transformers for language understanding. arXiv.org. Retrieved December 12, 2021, from https://arxiv.org/abs/1810.04805.