Time series analysis of COVID-19 related emotion detection using natural language processing

Chia-Wei Lin
lin.chiawe@northeastern.edu

Abstract

The COVID-19 outbreak which starts in early 2020 not only influences human's physical health, causes a rapid number of confirmed cases and death globally but also directly affects mental health issues worldwide. Social media like Twitter has been a valuable resource to gather public opinion with the help of natural language processing techniques and tools. This project aims to analyze the association between the global COVID-19 confirmed case number and the emotional response of English Twitter users over time. In order to obtain a reliable model for emotion detection, we build SVM, LSTM, bi-LSTM, pretrained-based BERT models on a labeled tweet datasets, then apply the best-performing model to a public twitter dataset which captured COVID-19 related tweets posted from January 2020 to classify tweets into four emotion, anger, fear, joy and sadness. Finally, we investigate the trend of emotions of tweets across time. Our result shows an improvement in accuracy by bidirectional LSTM model (accuracy = 0.48) compared with the baseline (accuracy = 0.37) on emotion classification. In addition, we detect that the fear related tweets dominated throughout the pandemic. Although the performance of our model is not outstanding, and the relation between emotions and COVID-19 new confirmed cases is not significant, we believe that the intention of this project generates some valuable insight on people's emotional reaction toward pandemic related issues and will aid in answering public health questions along with providing practical strategies for supporting the public.

1 Approach

1.1 Tweet Processing

Prior to training, the original corpus was cleaned from various symbols like @, , URLs and punctuation marks using the "re" python module. Emojis and unicode emoticons were handled and transformed into textual ASCII representations. All texts were converted to lowercase, tokenized, lemmatized and stemmed to obtain the original forms, with all English stop-words removed.

1.2 Word Embedding

1.2.1 FastText

To generate word embedding, we used pre-trained word vectors from fastText, an extension to Word2Vec proposed by Facebook in 2016. In specific, we used wiki-news-300d-1M.vec, which learned from Wikipedia 2017, UMBC web based corpus and statmt.org news dataset. We passed tokenized pre-processed text into the embedding matrix and outputted a 300-dimensional vector for each word.

1.2.2 GloVe

We also experimented on pre-trained word vectors from another well-known word2vec model, GloVe, an unsupervised learning algorithm performed on aggregated global word-word co-occurrence statistics from a corpus developed as an open-source project at Stanford. We used the 200-dimensional word vector trained on 2B tweets for the downstream tasks.

1.3 Models

1.3.1 Baseline, SVM with unigrams

We used the Support Vector Machine (SVM) classifier as our baseline model. Unigrams were vectorized with CountVectorizer in scikit-learn library and served as the input.

1.3.2 FastText Models

Since the fastText model can efficiently build scalable solutions for text representations and classification, we imported the fastText library, trained the classifier with default setting epoch=5 and

learning rate=0.05 on our training set and evaluated the performance.

1.3.3 LSTM and bi-LSTM Models

The Long Short-Term Memory (LSTM) deep neural network we designed contained multiple layers: a word embedding layer, LSTM / bidirectional LSTM layers, followed by a linear layer, which outputs a tensor of length 4 for each sample and returned the label with the maximal possibility. For word embedding features, we used GloVe to generate 200-dimensional word vectors as the input for downstream classification tasks. To be able to train the sequential neural network in batches, we normalized each tweet length by zero padding and took the mean of the words of the sentence as the output feature of the LSTM layer. We applied different dropout rates on the LSTM layers to prevent overfitting.

1.3.4 BERT Models

Bidirectional Encoder Representations from Transformers (BERT) is a new method of pre-training language representations which obtains state-of-the-art results of Natural Language Processing tasks including sentiment analysis and text classification. In our project, we used BERT Tokenizer along with BERT Embeddings as the input, then fine-tuned a English-only BERT-Small model on our classification task. We then added a single-layer classifier with the Sigmoid activation function, which received the output from the location of [cls] token and predicted the emotion category of a targeted tweet.

2 Experiments

2.1 Data

To assess the performance of different models on emotion classification tasks, we chose an annotated English twitter dataset as our train and development data. The dataset was released from Semeval-2018 Task 1: Affect in Tweets. The EI-oc dataset we used included 7102 training data and 1464 development data, labeled with one emotion category from anger, fear, joy and sadness. The original task for the dataset was to classify the tweet into one of seven ordinal classes of intensity of a given emotion E. However, we only used the emotion label for our classification goal. For the imbalanced data distribution, we applied class weight to the optimizer for the slight bias problem.

	Train set	Dev set
anger	1701	388
fear	2252	289
joy	1616	290
sadness	1533	397
total	7102	1464

2.2 Evaluation Method

To evaluate the performance of the models, we compared the accuracy on the dev set and generated confusion matrix for further understanding of how our model did on the classification job.

2.3 Experimental Details

2.3.1 Data preparation and word embedding

First we compared the performance of SVM models training on raw tweets's unigram with preprocessed tweets. We also ran the SVM models with fastText pre-trained word embedding and GloVe word embedding to evaluate the difference.

2.3.2 FastText

Since performing text classification with fastText is efficient and easy to implement, we also evaluated the performance of fastText with unigram or bigram.

2.3.3 LSTM and bi-LSTM

For all of our LSTM and bidirectional LSTM models, we used GloVe for the embedding matrix, class-weighted cross-entropy loss as our loss function and AdamW as the optimizer. Setting of 20 Epochs and a batch size of 64 was used for all models. First we compared the model using fixed embedding and unfixed embedding, then we compared the model using LSTM and bidirectional LSTM. As the unfixed word embedding and bi-LSTM combination outperformed the others, we trained over 60 models with different combinations of hyperparameters on it. The hyperparameters varied were hidden sizes, the number of bi-LSTM layers, and the dropout after the bi-LSTM layers. By training those models, we were able to explore the best model configuration. Once we determined which models outperformed, we ran another set of experiments with the best model to assess the best learning rates and different weight decay of the optimizer.

2.3.4 BERT

For the BERT model, we also used class-weighted cross-entropy loss as our loss function

and AdamW as the optimizer. 20 Epochs and a batch size of 64 was used for all models. Two different models were trained and evaluated, the first model had two linear layers followed by the pre-trained BERT layer and the second one had only one linear layer.

2.4 Results

The results of our first set of experiments are shown in the table below. Compared to the baseline model, SVM with raw unigrams, the SVM model with pre-processed (cleaned) unigrams had a better accuracy. Using word embedding from GloVe was better than from fastText. And for the fastText models, the one trained with unigrams outperformed the one trained with bigrams.

SVM w/ raw unigrams	**SVM w/ cleaned unigrams**
0.3716	0.34160
SVM w/ fastText	**SVM w/ GloVe**
0.4085	0.4112
fastText w/ unigrams	**fastText w/ bigrams**
0.4583	0.4372

Table: The accuracy of each SVM / fastText model

The results of second set of experiments are shown in the table below: Compared to the LSTM model, bidirectional LSTM model showed a better accuracy. Model with a trainable embedding layer outperformed the model with a fixed embedding layer.

LSTM w/ Train embedding	**bi-LSTM w/ Train embedding**
0.4094	0.4627
bi-LSTM w/ Fixed embedding	**bi-LSTM w/ Train embedding**
0.4504	0.4627

Table: The accuracy of each model

The table below displays the result of our bi-LSTM models which trained with different hyperparameters. We used preprocessed tweets as the input, GloVe word embedding as the initial weights for the embedding layer, several bidirectional LSTM as middle layers and a final linear layer for output, which was the best combination concluded from above experiment sets. As we expected, the model did not perform well when the model architecture was simple. As the hidden nodes and the number of layers increased, the model became more complex and the accuracy also increased. The best configuration was 256 hidden nodes, 3 layers with 0.0 dropout in our experiment, which reached 0.48 in accuracy. We also tried different learning rate and weight decay value for the optimizer, and the best setting was learning rate = 2e-3 and weight decay = 5e-4.

		Hidden Nodes / Number of Layers				
		32 / 1	32 / 2	32 / 3	32 / 4	32 / 5
Dropout	0.0	0.3958	0.4477	0.4559	0.3609	0.2686
	0.2	0.3917	0.4614	0.3103	0.3172	0.3117
	0.4	0.3944	0.3958	0.3985	0.3151	0.2891
		64 / 1	64 / 2	64 / 3	64 / 4	64 / 5
Dropout	0.0	0.3978	0.4662	0.3964	0.3903	0.3548
	0.2	0.3985	0.4668	0.3889	0.3527	0.2809
	0.4	0.3910	0.4491	0.4607	0.3315	0.3643
		128 / 1	128 / 2	128 / 3	128 / 4	128 / 5
Dropout	0.0	0.4163	0.4655	0.4498	0.4402	0.3739
	0.2	0.4033	0.4675	0.4539	0.3964	0.3390
	0.4	0.4019	0.4607	0.4662	0.4129	0.3554
		256 / 1	256 / 2	256 / 3	256 / 4	256 / 5
Dropout	0.0	0.4306	0.4518	**0.4785**	0.3992	0.3363
	0.2	0.4293	0.4662	0.4655	0.3930	0.2891
	0.4	0.4238	0.4655	0.3424	0.3527	0.2659

Table: The accuracy of each bi-LSTM model

Due to the resource and time constraints, we did not acquire any satisfied result on BERT models. The best model we trained only reached 0.26 in accuracy, which is very close to the probability of randomly guessing one class out of four.

BERT w/ 1 linear layer	**BERT w/ 2 linear layers**
0.2515	0.2604

Table: The accuracy of each BERT model

2.4.1 Quantitative analysis of the best model

By generating a confusion matrix for our best model, we gained a better understanding of how the model classified the data compared to the ground truth. The bi-LSTM model incorrectly predicted 33 percent of anger data and 33 percent of sadness data as fear, resulting in lower precision of the fear class. We also found that the f1-score of the joy class was higher than others, which implied that the model could classify the joy class more accurately. Given that only the emotion of joy has the property of positive sentiment polarity, but anger, fear and sadness are negative, the model

learned to classify emotions similar to the human's common sense toward these four emotions.

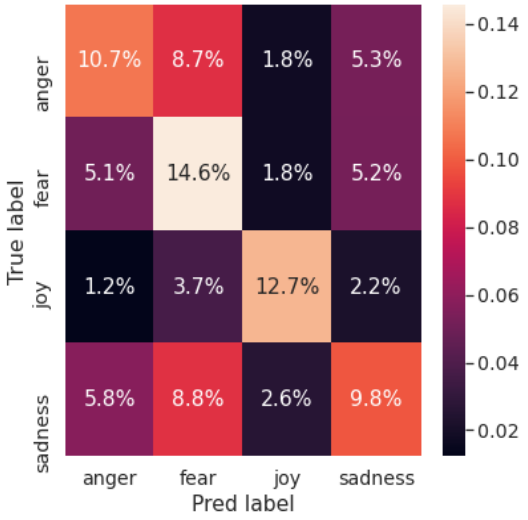


Figure: The confusion matrix of our best model

2.5 Time Series Tweets Analysis

2.5.1 Data

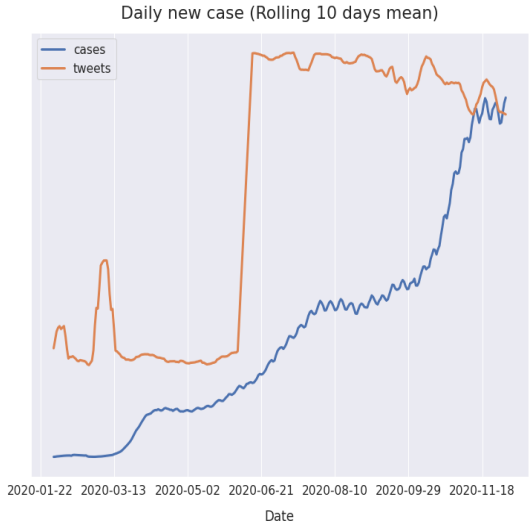
We used the COVID-19-TweetIDs dataset as our corpus, which is an open dataset capturing a collection of tweets ID using more than 70 different keywords and hashtags that are commonly used while referencing the COVID-19 pandemic. The dataset collected tweets posted from January 28, 2020 and updated daily, with 922,468,360 tweets ID in total so far. We hydrated the tweets ID dated from January 20 to December 4, 2020, and randomly selected 2000 English tweets per day to analyze the trend of emotion during the pandemic.

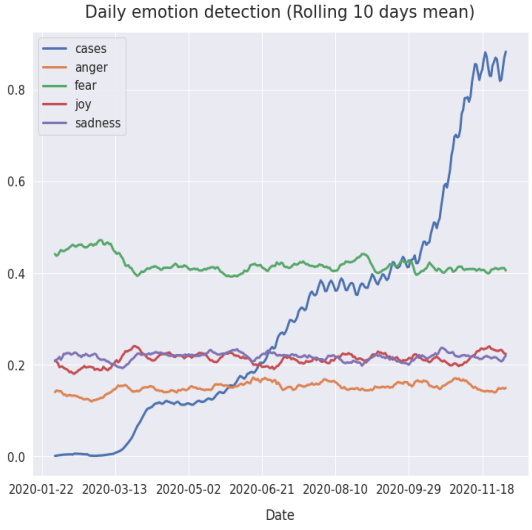
2.5.2 Experimental Details

We plotted the proportion of tweets in each emotion category for every 10 days with a line chart and compared it with the number of global COVID-19 new confirmed cases across time.

2.5.3 Results

In the first figure (daily new confirmed cases / number of tweets collected), we could see three obvious surges happened in early April, early August and mid November. Right Before the first and the second surge, the number of tweets collected by the dataset showed increases, then the number decreased after June even when the new confirmed cases reached a record high in November. If the collecting method had not been changed, we could state that the public raised attention and discussed about the pandemic in the very early stage of a surge but gradually became more calm and lost attention to the issue. The second figure (daily new confirmed cases / daily proportion of tweets stratified by emotion) showed that fear was the dominant emotion throughout the pandemic with no surprise. During the early pandemic, the proportion of fear had increased related to the emergence of COVID-19 and its unknown nature. However, the proportion of fear decreased in March and saturated afterward according to our result. The emotion of sadness and joy had similar proportions at all times and the anger remained the least with no significant changes.





3 Conclusion

3.1 Achievements

In this project, we evaluate through several approaches of emotion detection from tweets. First

we confirm that tweet preprocessing made much difference to the downstream classification goal, and the pretrained word embedding from large corpus also improves the performance of models. We observe that our best bidirectional LSTM model outperformed the baseline SVM model with accuracy increases from 0.37 to 0.48. Our error analysis reveals that the model does well on categorizing joy emotion but mislabeled 33 percent of anger and 33 percent of sadness to fear on development data. In the time analysis part, we also concludes that fear was the dominant emotion throughout the pandemic but slightly decreased after the first two months.

3.2 Challenges

From this project, I gain an understanding of how to design and conduct a research project from scratch. For me, the most difficult part is to narrow down and target the research interest from literature survey; the most important lesson I learned is that training a good-enough (not even approach to the state-of-the-art performance) machine learning model takes more time and resources than expected. I also become more familiar with big data processing and the implementation of building deep neural network models. However, there is less work on emotion analysis compared with sentiment analysis in text, and the amount of labeled dataset is limited also the annotation is somehow ambiguous by the nature of emotion. Besides, tweet messages are diverse and may include a rich ensemble of emotions, sentiments and moods. A single tweet may contain neutral emotion, no emotion or a combination of complex emotions. The usage of causal language and the limitation of text length are features of Twitter content. We detected around 20 percent of unrecognized words as 'inyourneighborhoodnotdc' or 'haappppppy' in the cleaned text. Above all increased the difficulty of emotion classification from tweets.

3.3 Future work

To further expand on our work, it would be interesting to test on different architectures of models, for example adding a convolutional neural network (CNN) layer for feature extraction, or use the output features from LSTM as the input of other classification models such as SVM. There are a large number of potential features that can be added along with word embedding like effective lexicon-based features or part-of-speech (POS) tagging to increase the model performance. Since the original training dataset also annotated the valence and emotion intensity, we could train different models, identify the tweets with low to high emotion intensity class and gain more insight of the emotion response in scale. The project could also be extended to apply topic modeling on top of current results for better understanding of public concerns.

4 Reference

1. Advances in Pre-Training Distributed Word Representations T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, A. Joulin. arXiv:1712.09405

2. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. Enriching Word Vectors with Subword Information, Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov, arXiv:1607.04606

4. Semeval-2018 Task 1: Affect in Tweets. Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. In Proceedings of International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, USA, June 2018.

5. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, arXiv:1810.04805

6. Chen E, Lerman K, Ferrara E Tracking Social Media Discourse About the COVID-19 Pandemic: Development of a Public Coronavirus Twitter Data Set JMIR Public Health Surveillance 2020;6(2):e19273 DOI: 10.2196/19273 PMID: 32427106

7. Documenting the Now. (2020). Hydrator [Computer Software]. Retrieved from https://github.com/docnow/hydrator

8. CrystalFeel at SemEval-2018 Task 1: Understanding and Detecting Emotion Intensity using Affective Lexicons, Gupta, Raj Kumar and Yang, Yinping, Jun 2018

9. COVID-19 Twitter Dataset with Latent Topics, Sentiments and Emotions, Attributes, https://arxiv.org/abs/2007.06954

10. Global Sentiments Surrounding the COVID-19 Pandemic on Twitter: Analysis of Twitter Trends, May Oo Lwin, PhD, Jiahui Lu, PhD, [...], and Yinping Yang, PhD