CS230

Sentiment Analysis of COVID-19 Tweets

Xixie Zhou
Stanford University
xixiez@stanford.edu

Abstract

In this article, two deep learning models with Logistic Regression and LSTM with two different word embedding techniques: TfidfVectorizer and CountVectorizer to study people's attitude towards COVID-19 when the global pandemic first took place by investigating their tweets burgeoning during that period. The best model was the LSTM model with TfidfVectorizer with f1 score upto 0.85.

1 Introduction

In the end of 2019, a global pandemic changed the world significantly. The new coronavirus or the COVID-19 is a contiguous disease caused by severe respiratory syndrome. The pandemic has altered people's life dramatically as the disease can be spread through inhalation. Even though vaccines are developed to prevent new people from getting infected, no effective therapies were found till now. Social media expressed substantial interest in this topic when the pandemic first burst out, and it might be inspiring to learn how people react to such a disaster when the U.S. government failed to take appropriate actions to prevent the spread of the COVID- 19. This project investigates thousands of COVID-19 related tweets and performs a sentiment analysis on people's reaction.

The input to the task consisted of two classes: the original tweets from the internet, and the manually labeled sentiment: negative, positive, etc.. Then in the featurization stage, the original tweets were pre-processed and vectorized using CountVectorizer and TF-IDF vectorization. In the modeling stage, Logistic regression and LSTM were implemented to classify an unknown data set. The general procedure was shown in Figure 1.

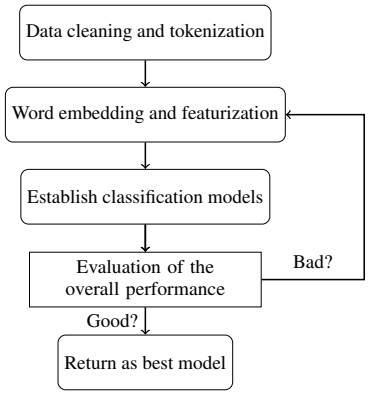

Figure 1: The general process of this task.

2 Related work

I focused on looking for related works that had the following property:

1. Performing sentiment analysis to a large number of reviews/tweets
2. Comparing different vectorization methods including CountVectorizer and TF-IDF
3. Comparing or fine-tuning different deep learning algorithms including Logistic Regression and LSTM

For Property 1, I found this article written by Vasu Jain [1] who pridicted whether a movie was success or not basued on sentiment analysis of tweets. He categorized all the labels into four categories, positive, negative, neutral and irrelevant. His categorization metric was interesting and provided me with new thoughts to imporve accuracy. Positive/negative/irrelevant were simple, but he defined his neutral to be 1) neighter positive nor negative, 2) mixed positive and negative, 3) unable to decide, 4) simple factual statements, 5) questions with no emotions, 6) hyperlinks. I thought this fully characterized the difficulty of manually labeling when the limit between positive and negative was vague, hence it could be a safer or conservative approach when categorizing sentiments. Another article that proposed an interesting idea of categorization was written by Xujuan Zhou et al. [2] who used Wilson opinion lexicon list to decide the words' semantic orientiations, where for each word +1 indicated positive and strong subjectivity, +0.5 indicated a positive and weak subjectivity, 0 indicated neutral, -0.5 indicated negative and weak subjectivity, and -1 indicated negative and strong subjectivity. They then added the scores for each tweet and placed them in to positive, negative and neutral. Similar to Vasu's article, even though it was relatively easy to categorize a certain tweet into positive, negative or neutral, it would be difficult to further distinguish between a 'strong' positive and 'weak' positive.

For Property 2, I found two articles that worth reviewing. Abinash, et.al. [3] used Naive Bayes model and SVM for sentiment classification of movies reviews. Even though the algorithm they used were different from this task, their detailed vectorization methods were worth reviewing. They applied two word embedding techniques to vectorize textual data: CountVectorizer and TF-IDF, and then fed the vectors to SVM and Naive Bayes model separately. In their case, the SVM with TF-IDF had a better performance, with accuracy as high as 94.06%. Similarly, In Yanfa and Masayu [4] developed a semantic model for Indonesian tweet categorization with TF-IDF and word2vec vectorization. The only difference is that they generated their own training data set instead of the pre-trained ones. The deep learning model they used were SVM and one-layer artificial neural network (ANN). The best result they achieved were 81.22% with one-layer ANN.

For Property 3, There was one article written by Karthik and Fathi [5] who applied LSTM model to analyze the 2016 presidential debate in Ohio. They developed six different parameter reduced slim LSTM models. The reason why they choose the simplified model was to speed up the training and reduce the computational cost. I believed this might be a good point since my data set contained over 40,000 tweets, and a deep LSTM model might consume too much time.

3 Dataset and Features

3.1 Data set

The data set collected tweets between 03/16/2020 and 04/14/2020 when the COVID- 19 became a societal topic, and most countries started to take actions. The data was provide by Aman Miglani on Kaggle [6]. The sorted data contains six classes: user name, screen name, location, tweet date, original tweet, and a label. In this analysis, I used last two classes to investigate people's attitude towards COVID-19. The distribution of the manually labelled sentiment was shown in the pie chart, which was further grouped into 3 categories for more precise classification. The reason behind this was because the limit between positive/extremely positive was quite vague, and so did negative/extremely negative. The distributions were shown in Figure 1.

3.2 Pre-process

At the very first step, the sentiment column was converted to integer labels. Hence this task was treated as a multi-label classification problem. We could see that neutral was 18.7%, positive the

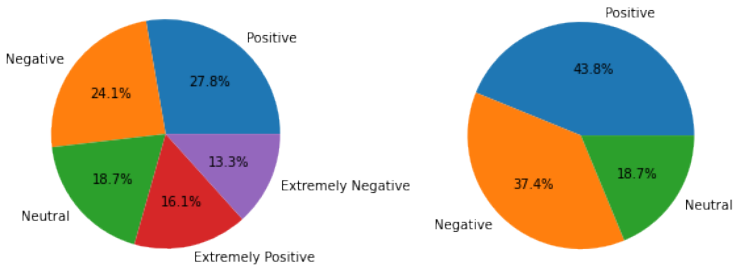

Figure 2: Distributions of the sentiments.

highest was 43.8% and negative the lowest was 37.4%. This indicated most people had an positive attitude when the pandemic broke out.

In the pre-processing stage, the data was split into 41157 training samples and 3798 testing samples according to the author, and among the training samples, 10% of the stratified data were split into dev set, which was used for fine-tuning in later stages. As a typical procedure in NLP problems, only meaningful words were counted, hence all hashtags, mentions, hyperlinks, punctuation were removed from the original tweets. All stop words were removed as well, and non-stop words were returned to their stem form and lower cased. A few samples of pre-processed sentences were shown in Figure 3. Last, to ensure the input vectors were well centered, L2-norms were applied in all layers.

```
0
1    advice talk neighbour family exchange phone nu...
2    coronavirus australia woolworth give elderly d...
3    food stock one empty please panic enough food ...
4    ready go supermarket outbreak paranoid food st...
5    news region first confirmed covid case came su...
6    cashier grocery store sharing insight prove cr...
7                  supermarket today buy toilet paper
8    due covid retail store classroom atlanta open ...
9    corona prevention stop buy thing cash use onli...
Name: Corpus, dtype: object
```

Figure 3: A few samples of pre-processed data.

3.3 Features

In this task, two featurizations were implemented: CountVectorizer and TF-IDF [7] vectorization. CountVectorizer is a simple vectorization method provided by the scikit-learn library, which simply vectorized a given text based on the count of each word that occurs in the entire corpus. It transformed the input text to a token count matrix. TF-IDF stood for term frequency0inverse document frequency, similar to the CountVectorizer, offered us statistically how important a word was given to a document in corpus. The weights were given by

$$W_{t,d} = \text{tf}_{t,d} \times \text{idf}_t, \tag{1}$$

where $\text{tf}_{t,d}$ represented the ratio of word t in a document d to the total words in the document, and idf_t is the logarithm of the ratio # documents with word to the total documents in corpus.

4 Methods

Two models were implemented to handle this task, the Logistic Regression and LSTM. Here I gave some of the highlights of the two algorithms.

4.1 Logistic Regression

The Logistic Regression is essentially a one layer neural network which applies weights and biases for each input feature, in the first step in the hidden layer:

$$A = WX + B \tag{2}$$

where A, W, X, B are all vectors. Then in the second step a sigmoid function was applied such that

$$h(A) = \frac{1}{1 + e^{-A}}, \tag{3}$$

the parameter W, B can be updated by gradient descent using backpropagation via

$$\theta := \theta - \alpha \frac{\partial J}{\partial \theta} \tag{4}$$

where α is the learning rate, and J is the cost function given by summing up individual losses

$$J = -\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log\left(a^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - a^{(i)}\right) \tag{5}$$

where m is the number of samples in the training set.

4.2 LSTM

The LSTM model belongs to the family of recurrent neural networks. The following graph created in colah's blog [8] might be helpful in understanding the basics of LSTM. At each time stamp t, the

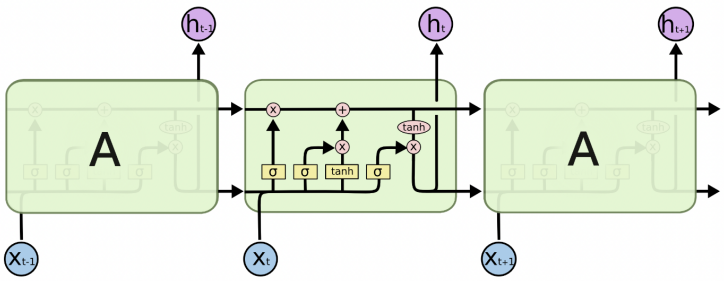

Figure 4: The graphical overview of an LSTM model.

model calculates the following [9]:

$$\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_i i + W_{hi}h_{t-1} + b_{hi}) && (6)\\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) && (7)\\
g_t &= \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) && (8)\\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) && (9)\\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t && (10)\\
h_t &= o_t \odot \tanh\left(c_t\right), && (11)
\end{aligned}$$

where i_t, f_t, g_t, o_t are the input, forget, cell, and output gates, h_t is the hidden state at time t, c_t is the cell state at time t, x_t is the input at time t, σ is the sigmoid function, $\odot$ is the Hadamard product.

5 Experiments/Results/Discussion

For the preliminary analysis of the data, the multi-class logistic regression algorithm was implemented as the baseline model. I chose the logistic regression algorithm to be the baseline model because of its simplicity and low requirement of computation. The featurization method here was TfidfVectorizer. The training/dev set were split with a ratio 9:1. The average accuracy of the logistic model was about 0.55, and the average F1 score was 0.53 for the five-label classification, and 0.79 average accuracy,

Accuracy	5-label with LR	3-label with LR
CountVectorizer	0.57	0.77
TfidfVectorizer	0.53	0.79

Table 1: Accuracy for the Logistic Regression algorithm.

F1 score	5-label with LR	3-label with LR
CountVectorizer	0.59	0.76
TfidfVectorizer	0.55	0.79

Table 2: F1 score for the Logistic Regression algorithm.

0.79 F1 score for the three-label classification. I further more used the CountVectorizer featurization and the best results are shown in the following two tables.

The best result with LSTM/TfidfVectorizer was shown in Table 3. The learning rate here is 0.01. This LSTM model contained four hidden layers, an TF-IDF embedding layer, an LSTM layer with 200 units, a dropout layer with 0.2, and a dense layer with 3 units for the 3-label and 5 units for the 5-label. The optimizer here was Adam, one of the most commonly used optimizer in deep learning. The train/dev split was 8:2. Early stopping was applied to prevent overfitting and to save time. The whole training process took about 4 hours with properly tuned batch size = 300 and epochs = 80. The bias error was quite low with accuracy about 0.97 in the training set. The precision for negative/neutral/positive are 0.87/0.80/0.85 respectively.

TfidfVectorizer	5-label with LR	3-label with LR
Accuracy	0.72	0.82
F1 score	0.74	0.85

Table 3: Metric for the best LSTM algorithm.

the confusion matrices for results in Table 3 are

```
array([[379,   0, 188,   8,  17],
       [  2, 449,  23,   2, 123],
       [ 77,  14, 751,  74, 125],
       [  0,   1,  50, 518,  50],
       [  9,  86, 104,  36, 712]])
```

```
array([[1361,   92,  180],
       [  54,  508,   57],
       [ 156,   34, 1356]])
```

6 Conclusion/Future Work

This task was accomplished by using Logistic Regression and LSTM deep learning models along with two word embedding methods: CountVectorizer and TfidfVectorizer. The two models were tuned properly in terms of batch size, epochs, hidden units and several other hyper-parameters. The best result was achieved using the LSTM/TF-IDF model. The F1 score was 0.85.

Overall the task was successfully accomplished. If more time is allowed, it would be interesting to investigate the 5-label classification. By relabeling the 5 sentiments using methods similar to Xujuan Zhou's article, I believe the ambiguity between positive and extremely positive, negative and extremely negative might be quantified properly.

References

[1] J, Vasu. "Prediction of Movie Success Using Sentiment Analysis of Tweets." The International Journal of Soft Computing and Software Engineering [JSCSE], vol. 3, no. 3, Mar. 2013.

[2] X, Zhou, et al. "Sentiment Analysis on Tweets for Social Events." IEEE 17th International Conference on Computer Supported Cooperative Work in Design, June 2013.

[3] A. Tripathy, A. Agrawal, S. K. Rath, "Classification of Sentimental Reviews Using Machine Learning Techniques", Procedia Computer Science, Vol. 57, 2015, pp. 821-829.

[4] Y. A. Putra and M. L. Khodra, “Deep learning and distributional semantic model for Indonesian tweet categorization”, 2016 International Conference on Data and Software Engineering (ICoDSE), Denpasar, 2016, pp. 1- 6.

[5] K. Gopalakrishnan, F. M. Salem, “Sentiment Analysis Using Simplified Long Short-term Memory Recurrent Neural Networks", arxiv, 2020.

[6] M, Aman, “Corona Virus Tagged Data,” Kaggle, https://www.kaggle.com/datatattle/covid- 19-nlp-text-classification.

[7] G. Salton, A. Wong, and C.-S. Yang, “A Vector Space Model for Automatic Indexing”, Communications of the ACM, Vol 18, no. 11, 1975, pp. 613–620.

[8] Colah’s blog, “Understanding LSTM Networks", 2015.

[9] LSTM — PyTorch master documentation.