COVID-19 Tweet Sentiment Classification

Zach Zhu

Abstract

This project is leveraging multi-class text classification algorithms (Logistic regression, Random Forest, FastText, and SVM) to predict sentiment of Tweets related to COVID-19 on labelled open-source data. The best is a FastText model with 0.7829 F1 score that can be productized as a REST service. **Please refer source code here.**

1 Introduction

The main theme of the year 2020 is the pandemic. The COVID-19 pandemic has pretty much shattered the world and is discussed whether online or offline. The project aims to understand how the general public feel about this pandemic: are they feeling positive, negative or neutral. It would be beneficial to make prediction for other similar situations, and better inform policy makers.

2 Related Work

Machine learning for NLP and text analytics is not really a new thing but has been a hot topic ever since the booming of the social media as massive online text data coming in every day. Many researchers have been working on the Multiclass classification to process and analyze text corpus.

As we know, Multiclass or multinomial classification is the problem of classifying instances into one of three or more classes, which is an important part of machine learning.

Support vector machines (SVMs) is a supervised learning can be used for classification and was developed at AT&T Bell Laboratories by Dr. Vapnik and his colleagues[1]. Rennie and Rifkin (2001)[2] compare Naive Bayes and Support Vector Machines on the task of multiclass text classification and find that SVMs substantially outperform Naive Bayes.

FastText is a library for learning of word embeddings and text classification created by Facebook's AI Research (FAIR) lab[3]. Joulin et al (2016)[4] explore a simple and efficient baseline for text classification and the experiment showed that the fast text classifier FastText is often on par with deep learning classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation. Alessa et al (2018)[5] used FastText to classify Twitter posts into flu-related or flu-unrelated posts and results show that the framework improves the accuracy.

3 Dataset

The data for this project is obtained from Kaggle which is an open-source dataset. The data contains two csv files: Corona_NLP_test.csv and Corona_NLP_train.csv. Corona_NLP_train.csv will be used for training the NLP models and Corona_NLP_test.csv will be used for evaluation.

[1] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks". Machine Learning. 20 (3): 273–297.
[2] Rennie, Jason D. M.; Rifkin, Ryan. (2001). “Improving Multiclass Text Classification with the Support Vector Machine”. AIM-2001-026CBCL-210
[3] https://research.fb.com/blog/2016/08/fasttext/
[4] Joulin, Armand; Grave, Edouard; Bojanowski, Piotr; Mikolov, Tomas. (2016). “Bag of Tricks for Efficient Text Classification”. arXiv:1607.01759 [cs.CL]
[5] Alessa, Ali; Faezipour, Miad; Alhassan, Zakhriya. (2018). "Text Classification of Flu-Related Tweets Using FastText with Sentiment and Keyword Features," 2018 ICHI, New York, NY, 2018, pp. 366-367, doi: 10.1109/ICHI.2018.00058.

Dataset Fields:

1) Location
2) Tweet At
3) Original Tweet
4) Label

Training data contains 41,157 records of data, and testing data contains 3,798 records. The data is very neat and clean not required much data cleansing but still needed a few steps to fit the text classification model. First, check missing values and label classes. Only location contains missing values, but we don't require that information. The label classes are that inbalanced and will not require further process.

Labels	Proportion
Positive	0.277523
Negative	0.240955
Neutral	0.187404
Extremely Positive	0.160945
Extremely Negative	0.133173

For the tweet corpus, all URLs are removed since they don't really provide any sentiment information. After that, each tweet is tokenized to get single word, and all numbers, punctuations, stop words are removed. 120 records of text not containing any words are also dropped. Finally, 41037 records of text with average 91 words are used for modelling.

4 Method

Four machine learning algorithms: Logistic regression, SVM, FastText, and Random Forest will be implemented to solve this task. In each algorithm, will compare different sets of parameters and choose the best model with the highest F1 score. Before fitting data in the model, text will be vectorized and transformed into a TF-IDF matrix to better adjust the weight for the importance of each word.

4.1 TF-IDF[6]

TF-IDF (term frequency–inverse document frequency), is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. [7] This is done by multiplying two metrics: times a word appears in a document, and the inverse document frequency of the word across a set of documents. Both unigrams and bigrams will be considered in the transform. The feature matrix will be used for logistic regression, SVM, and random forest, while FastText requires data to be in the format of ": __label__1 text" so not using the transformed feature vector.

4.2 Logistic Regression

Logistic regression model could be easily implemented in Python and can be used for both binary classification and multiclass problem. The model results will be considered as the benchmark compared to other algorithms.

Parameters	F1 Score
C=2, penalty='l1', solver= 'liblinear'	0.5210
C=15, penalty='l2', solver='liblinear'	0.4714
No penalty, solver= 'lbfgs'	0.5004
C=10, penalty='l2', solver='lbfgs'	0.5102
C=15, penalty='l2', solver='lbfgs'	0.5036

There are three parameters tuned for the model, penalty including l1 (Lasso) and l2 (Ridge), and C **(**Inverse of regularization strength, smaller values specify stronger regularization), and solvers ('lbfgs' and 'liblinear'). Lasso will shrink some coefficients to zero while ridge won't. One thing worth to note is that 'liblinear' is limited to one-versus-rest schemes while 'lbfgs' handles multinomial loss. The model with C=2, penalty='l1', solver='liblinear' gave the best result with 0.5210 F1 score. For solver lbfgs', it seems like with stronger Ridge regularization, the model performs better. For solver 'liblinear', with stronger Lasso regularization, the model performs better.

4.3 FastText

For FastText, two hyperparameters are tuned for the task. One is wordNgrams which is max length of word ngram, and the other one is learning rate. When learning rate is increasing, it can converge more quickly but might skip the local optimum and fail to converge.

[6] Spärck Jones, K. (1972). "A Statistical Interpretation of Term Specificity and Its Application in Retrieval". Journal of Documentation. 28: 11–21.

[7] Rajaraman, A.; Ullman, J.D. (2011). "Data Mining" (PDF). Mining of Massive Datasets. pp. 1–17. doi:10.1017/CBO9781139058452.002. ISBN 978-1-139-05845-2.

Parameters	F1 Score
lr=0.1, wordNgrams=1	0.7829
lr=0.1, wordNgrams=2	0.7642
lr=0.2, wordNgrams=2	0.7629
lr=0.5, wordNgrams=2	0.7586

It turns out the default model gives the best result with F1 score = 0.7829.

4.4 SVM

For SVM, three hyperparameters are tuned:
1. Loss function: 'hinge' is the standard SVM loss while 'squared_hinge' is the square of the hinge loss.
2. Penalty: l1 (Lasso) and l2 (Ridge)
3. C: Regularization parameter

Parameters	F1 Score
penalty='l2', C=1, loss='hinge'	0.4730
penalty='l2', C=10, loss='hinge'	0.4495
penalty='l2', C=10, loss='squared_hinge'	0.4727
penalty='l1', C=1, loss='squared_hinge'2	0.4996

In this case, SVM models are not doing so well compared to Logistic and FastText. The best model with gives penalty='l1', C=1, loss='squared_hinge'2 gives highest F1 score (0.4996).

4.5 Random Forest

Compared with previous research of this dataset, I suspect that the tree model may not perform that well for text classification. To be more rigorous, I conducted grid search and used cross validation to tune the hyperparameters.
1. max_depth: the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than minimum number of samples required to split an internal node. 10 or 30 or 50
2. n_estimators: The number of trees in the forest. 200 or 500.

Usually, the number of trees and depth of the tree are the most two important hyperparameter we tune for a tree model. The best model gives only 0.4289 in F1 score with max_depth=50 and n_estimators=200.

5 Result

To compare different algorithms and select the best model, I choose F1 score to be evaluation metric and the reason is that F1 score is the harmonic mean of the precision and recall that is a better metric when there are imbalanced classes (Label are not that equally distributed).

Algorithm	F1 Score
Logistic	0.5210
FastText	0.7829
SVM	0.4996
Random Forest	0.4289

Compared with other machine learning models, FastText won with a landslide victory. The F1 scores are very similar around 0.5 for others, while FastText achieved almost 0.8.

6 Discussion

The project only implemented four machine learning algorithms for the text classification and there are many limitations regarding the hyperparameter settings for each model. Supervising, the SVM performs even more poorly than Logistic which is unusual based on previous literature research and related work. One reason could be that the SVM need to be further tuned on different parameter settings. In addition, there are definitely a lot more other good algorithms worth trying for this task, such as Naive Bayes classifier, XGBoost, and neural network (CNN, RNN, LSTM).