Sentiment Analysis on Yelp's reviews Machine Learning for Natural Language Processing 2020

VU Thi Van Anh
`thivananh.vu@ensae.fr`

Abstract

This project illustrates a comparison among predicting methods used in *Sentiment analysis*. It includes several classic models ranging from fully connected NN, Random forest, gradient boosting and XGB using with TF-IDF features and other features; LTSM model with Glove embedding layer and finally, BERT.

1 PROBLEM FRAMING

Sentiments relates to feelings, attitudes, emotions and opinions which are the key influencers of human behaviours. Coincided with the explode of social media, industrial activities surrounding sentiment analysis have thrived, especially in catering and hospitality industry.

This project is conducted to analyze customers sentiments or satisfactions on using catering and hospitality services nearby. This project involves the multiple classification task of separating positive, negative and neutral comments.

2 EXPERIMENT PROTOCOLS

A. Dataset

Dataset contains reviews comments and rating stars left on Yelp website and application, which are available on Yelp challenge website. While this dataset requires creating account to download and it's quite heavy (5-6GB), so another csv file has been created by extracting 20,000 comment lines from the original dataset, and this reviews csv files is used in this project.

In this classification problem, rating stars results has been used as label. While rating scale ranging from 1 to 5 so, we decided all comments rated with 1 and 2 stars as negative, rated with 3 stars as neutral and the others with 4 and 5 stars as positive.

B. Implementation

The implementation framework is built with Tensorflow 2.2.0 and available on Github [1]. It requires dataset and a conda environment and runs entirely from a command line. A file *reviews.csv* stores 20,000 comment lines extracted from the raw dataset has been used in this project.

3 MODEL ARCHITECTURES

A. Classic models with TF-IDF features

TF-IDF feature - After cleaning text (by using tokenizing, lemmatizing, removing stopwords), TF-IDF is applied to determine a corpus of dataset (including 4446 words) and to vectorize words. TF-IDF calculates values for each word in a document through an inverse proportion of the frequency of the word in a particular document to the percentage of documents the word appears in. Although TF-IDF is an efficient and simple algorithm for matching words, it has limitations, especially in terms of synonyms while it does not make the jump to the relationship between words.

Other statistical features - In this section, several features are also considered such as number of characters, number of words, number of different words, number of digits, number of non-words and average characters per word.

Classic models including fully-connected NN with 2 hidden layers (inner layers are activated by ReLU function and the last layer passes through the softmax function), Random forest, Gradient boosting, XGBoost and LightGBM have been employed. In this section, we ignored the tuning hyperparameters part due to the time constraint limited resources.

[1] `https://github.com/vuthivananh43/ENSAE_NLP_2020`

B. LSTM model with GloVe embedding layers

In this part, I restricted the reviews to the top 20000 most common words and cut off the reviews with maximum length of 250 words. In Keras, the embedding matrix is represented as a ”layer”, and maps positive integers (indices corresponding to words) into dense vectors of fixed size (the embedding vectors). GloVe 300-dimensional vectors [2] has been employed in this part. Because our training set is quite large, we will update the word embeddings instead leave their values fixed (fine-tuning).

A recurrent neural network such as LSTM is used here while word embeddings pass in a multi-layer LSTM, which returns the last LSTM output state for each sequence. Each sequence is then fed into 2 hidden layers bi-directional LSTM, then passed to a (eventually multi-layer) dense network, with final dimension vocabulary size. While the inner layers are activated by a leaky ReLU function, the last layer of the dense block passes through the softmax function.

C. BERT model

BERT is conceptually simple and empirically powerful. BERT (Bidirectional Encoder Representations from Transformers) is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

4 RESULTS

This part reports the training results for $n = 30$ epoches for classic NN model, learning rate of 0.1 in Gradient boosting methods; $n = 15$ epoches in LSTM model with learning rate of 0.001. The hyperparameters were empirically fixed to embedding size= 300, hidden size LSTM = 128, number of LSTM layers = 2, dropout = 0:1 during training, and for the BERT model with $n = 5$ epoches.

[2] https://nlp.stanford.edu/projects/glove/

Methods	**Train set**	**Test set**
TF-IDF and Neural network	86%	80%
TF-IDF and Random Forest	99.99%	77.6%
TF-IDF and Gradient boosting	87%	80.2%
TF-IDF and XGBoost	87.2%	80.8%
TF-IDF and LightGBM	85.9%	80.1%
GloVe 50 and LSTM	80%	79.2%
BERT	91.8%	86.2%

5 CONCLUSION

As can be seen on the results above, TF-IDF features seem to be a quite efficient tool in Sentiment analysis even it’s limitations on capturing text positions or semantics. By combining with other features, the use of TF-IDF feature give the same significant results in several classic models, approximately 80% on test sets (except for Random Forest model which suffers from high overfitting problem).

On the other hand, it’s obvious that the LSTM model is not much powerful in categorizing opinions expressed in a piece of text while the result does not much improve (remain at 80% on test set). Even LSTM is a very powerful technique, it seems that this model is appropriate to be used in sequential textual data where the meaning of word depends on the ones that preceded it (in the translation technique) than in Sentiment Analysis where words frequency plays more important role.

In the last part, BERT model, a powerful deep learning algorithm for natural language processing, gives significantly higher performance than previous models even it does not require much training. By applying BERT, we obtained high significant accuracy of 86% on test set versus 91.8% on train set.

Finally, in this project, I ignore to tune-hyperparameter in models (we may obtain much better results if doing tuning models) due to time limits and limited computational resources.