Text Binary Classification Report: In-class Kaggle Competition

Xiuping Hua 30 March 2020

In this competition, we needed to perform text binary classification using feature engineering and model construction. After data preprocessing like noise removal, I did feature engineering including basic text feature extraction and word to vector representation. Then I built simple models, ensemble models as well as deep learning models. Based on model evaluation, Logistic Regression was the best fit in this case with the highest test accuracy of 74%.

1. Data description

In training dataset, there were 44183 samples, of which 24992 were labelled with 1 and 19191 were labelled with 0. The sizes of two classes were a little bit unbalanced, but it might be consistent with the real situation. In testing dataset, there were 28200 samples that we needed to do prediction.

2. Data Preprocessing

Data Preprocessing is the process of cleaning and preparing text for classification. The texts in datasets contained lots of noise, uninformative parts such as HTML tags, and professional expression such as formulas, equations and codes. Here was the hypothesis of having the data properly pre-processed should help improve the performance of the classification.

Table 1: Procedures of Data Preprocessing

Lowercasing	use string lower() function, converting uppercase characters into lowercase characters.
Punctuations Removal	use dict.fromkeys() function, mapping punctuation to None.
Stopwords Removal	use English stopwords corpus in NLTK and remove stopwords from texts.
Number Removal	use regex expression '\d' to find digits in texts and remove them.
Single Letters Removal	keep words with length longer than 1.
Lemmatization	use NLTK Lemmatizer by taking POS tag into account to convert a word into its base form.
Whitespace Removal	join all words together to remove extra whitespace.

Since there were a lot of professional expressions, such as formulas, equations and codes, removing punctuations and numbers might lead to information loss. And it turned out that data preprocessing removed important information after comparing the performance of same model with original text and with preprocessing text.

3. Feature Engineering

For a machine to make sense of natural language, it needs to be converted into some sort of a mathematical framework which can be modelled. In this competition, I firstly extracted some basic features of texts with labels 0 or 1, separately. Then I converted text into vectors using TF-IDF.

3.1 Basic text feature extraction

I extracted basic text features to see whether there existed differences between texts with label 0 and 1. Hypothesis was that texts with label 1, i.e., getting more votes, were more readable, providing more details with longer length of sentences, etc.

Table 2: Basic Features of Texts

Words_count	total number of words in each text.
Characters_count	total number of characters in each text.
Average_words	average number of words for sentences in each text
Top_words_count	number of most popular words in each text.
Least_words_count	number of least popular words in each text.
POS_tags_nouns/pronoun/verb/adv/adj	number of noun/pronoun/verb/adv/adj words in each text.

However, there was no obvious difference about above features between texts labelled with 0 and 1. So I did not use these basic features as the input of models. Details are shown in Appendix I.

3.2 Vector Representation

- **N-gram:** I used a sequence of N words and N characters separately to represent sentences of texts instead of just singular words or characters. The number of N had been turned and the one with best performance was chosen. This helped to boost accuracy.
- **TF-IDF:**TF-IDF is a very popular and standard tool in text classification. TF-IDF score represents the relative importance of a term in the document and the entire corpus. I created both word level TF-IDF and character level TF-IDF to represent texts in dataset.

I created TF-IDF for original texts and texts with data preprocessing as input of models. TF-IDF matrix from original texts made model achieve better result, and this might be because that it kept most of information. So all models used TF-IDF matrix from original texts as input.

4. Training/validation split

I split the training data into training dataset(80%) and validation dataset(20%). Validation dataset was used to compare the performance of models and to choose the best model. After choosing the best model and tuning parameters, I used all training dataset for training model.

5. Model Building

I trained different models from simple classification models to ensemble to models, and finally tried deep learning models to see whether they can find some deep connections in texts. For easier optimization of each model, the classification is done with pipelines like GridSearchCV. Finally, different metrics are examined for the model evaluation: accuracy, precision, recall and roc_auc. Details are shown in Appendix II.

5.1 Simple Classification Models

Logistic regression

In natural language processing, logistic regression(LR) is usually the baseline supervised algorithm for classification. LR is used to classify an observation into one or two classes, like 'positive sentiment' and 'negative sentiment'. In this case, LR is the best model with the highest test accuracy of 74% after parameter tunning.

Support Vector Machine

The support vector machine(SVM) is a supervised learning method. It determines the best decision boundary between vectors. So provided that I created vector representations which encode information from texts, I applied SVM algorithm to text classification. The biggest problem was it took a long time to train model.

Multinomial Naïve Bayes

The multinomial Naïve Bayes (Multinomial NB) is suitable for classification with discrete features, like word counts for text classification. In this case, I used a fractional count TF-IDF to train model.

5.2 Ensemble Models

XGBoost

XGBoost is an implementation of gradient boosted decision trees. It is an ensemble technique where new models are added to correct the errors made by existing models. It dominates structured datasets on classification and regression prediction modelling problems. However, the performance of XGBoost did not help improve accuracy.

LightGBM

LightGBM is based on decision tree algorithm and it splits tree leaf wise. So it can result in less loss and better accuracy. Also, it is surprisingly fast. Usually, LightGBM is used for ranking, classification and other machine learning tasks. So I tried this algorithm and achieved a relative good test score.

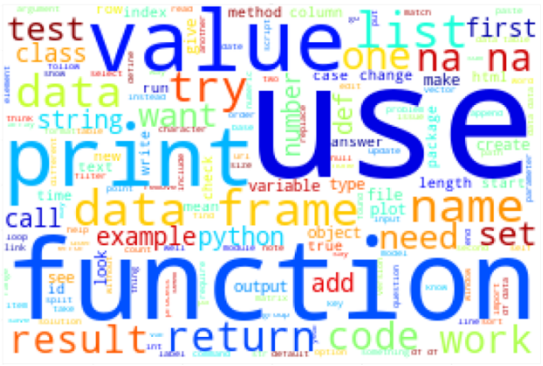
5.3 Deep Learning

CNN

CNN is a class of deep, feed-forward artificial neural networks. Each hidden layer will detect a special pattern of texts. Also CNN can identify patterns in the sentence regardless of their position, which means position invariant. However, in this case, I created a 3 hidden layer model and this model was overfitting with an validation score of 68%.

Appendix 1

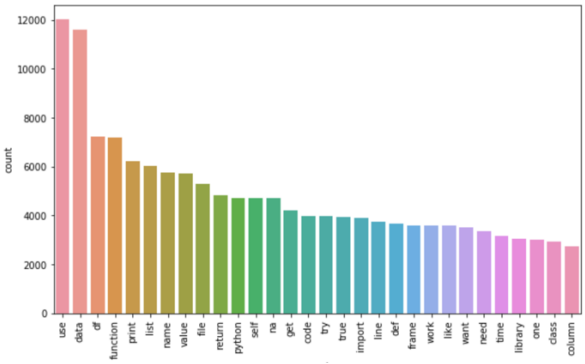
Word Cloud for Label 0



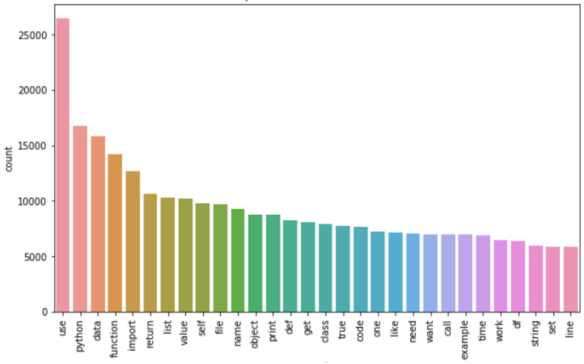
Word Cloud for Label 1

Picture 1: Important words for Label 0 and 1 are similar.

Top 30 words for Label 0

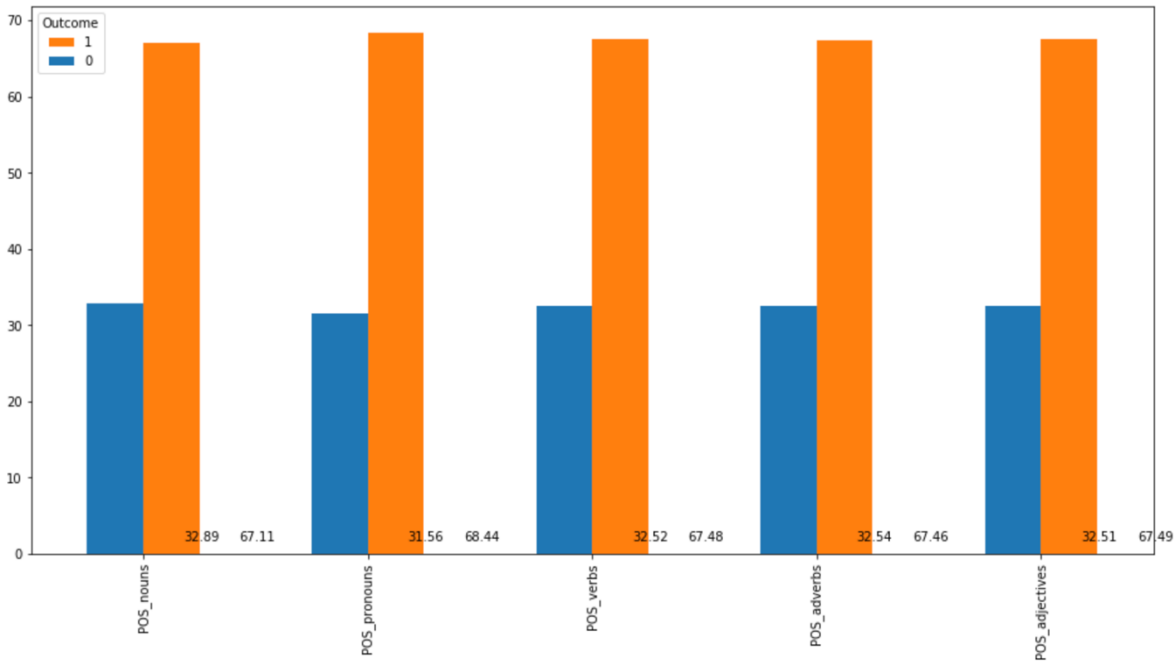


Top 30 words for Label 1



Picture 2: Most frequent words for Label 0 and 1 are similar.

POS for Label 0 and Label 1



Picture 3: Distribution of POS for Label 0 and 1 are similar.

Appendix 2

Model Parameters Tuning and Performance Comparison

	Model Name	Parameters Tunning	Best Parameters	Training Score	Validation Score	Training time
Simple Classification Models	Logistic Regression *	'C': [0.01, 1, 3]	'C': 1	0.87	0.74	30.4
	SVM	'C': [0.1, 1], 'gamma':[0.5, 0.9]	'C': [1], 'gamma':[0.5]	0.83	0.71	Very long ☹
	Multinomial NB	'alpha': [0.0001, 0.01, 0.5, 0.95]	'alpha': 0.01	0.78	0.69	1.5
Ensemble Models	XGBoost	'min_child_weight': [1,3,5], 'gamma': [0.5, 1, 2], 'subsample': [0.6, 1], 'colsample_bytree': [0.6, 0.8], 'max_depth': [5, 7, 9], 'learning_rate': [0.01, 0.1, 0.5]	min_child_weight': 1, 'gamma': 2, 'subsample':0.6, 'colsample_bytree': 0.6, 'max_depth': 9, 'learning_rate': 0.1	0.72	0.69	15064.3
	LightGBM *	'max_depth': [5, 7, 9], 'learning_rate': [0.01, 0.1, 0.5]	'max_depth': 9, 'learning_rate': 0.1	0.77	0.71	8602.2
Deep Learning Models	CNN		3 hidden layers	0.98	0.68	623.7

* Selected as final models for top 2 submissions.