Tweet Collection Dataset Classification Problem

Fatemeh Ahmadvand
Politecnico di Torino
s301384
Torino, Italy
s301384@studenti.polito.it

MohammadSadegh Radmehr
Politecnico di Torino
s301623
Torino, Italy
s301623@studenti.polito.it

***Abstract*—Mining social network data from unstructured and informal data is a challenging task. In this report we introduce a possible approach to the Tweet Collection Dataset classification problem. This study aims to measure the accuracy of the sentiment analysis classification model by using Logistic Regression classifier for multinomial models to extract the sentiment of tweet reporting in the feature. The proposed system was tested on a dataset of almost 80k tweets and was able to classify up to 78.2% of tweets accurately.**

I. Problem Overview

Sentiment analysis refers to identifying as well as classifying the sentiments that are expressed in the text source. Tweets are often useful in generating a vast amount of sentiment data upon analysis. These data are useful in understanding the opinion of the people about a variety of topics.

In this project, we carried out using two datasets in csv format, one labelled sentiment for training the model and one without sentiment labels, evaluation dataset. we need to analyze sentiments of text in the development dataset and then predict the sentiment for the evaluation dataset. The development dataset has 224993 unique and non-empty entries and contains six columns: sentiment, ids, date, flag, user and text. The evaluation dataset has 74998 entries, which have to be classified. The text column contains the tweet text itself and sentiment represent if the tweet text is positive or not, the values reported are 1 and 0.

II. Proposed Approach

A. Data exploration

In the first place, according to Fig.1 , it is vividly seen that the positive number is more than the negative ones. Therefore, our dataset is imbalanced. One of the ways at which we selected to deal with the imbalanced dataset is by resampling with sklearn.resample i.e. upsampling the minority class or downsampling the majority class.

In the second place, after extracting words, we gave the raw data to the word cloud. Word Cloud is a data visualization technique used for representing text data in which the size of each word indicates its frequency or importance. So, by using this technique we found out that which word is repeated more in the text Fig.2 and Fig.3.

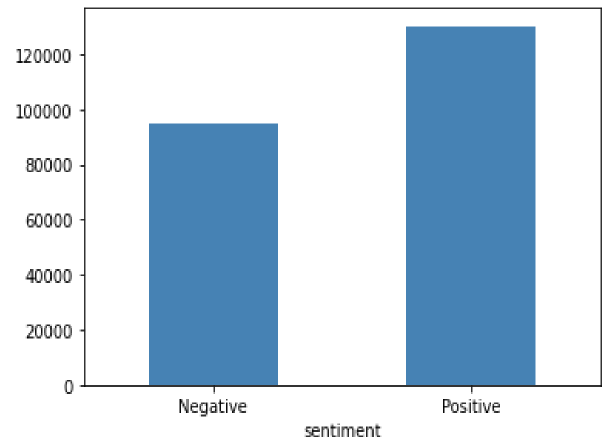

Fig.1: number of sentiment in development dataset

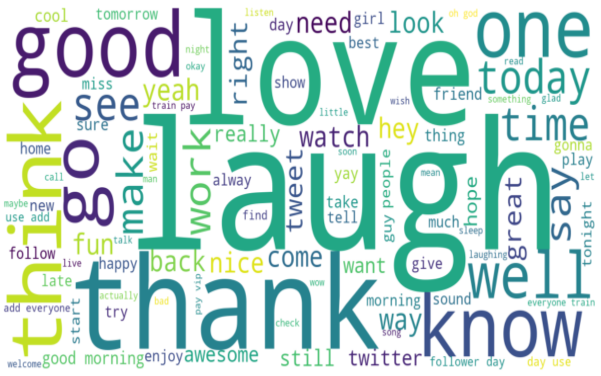

Fig.2: Word Cloud for positive sentiment

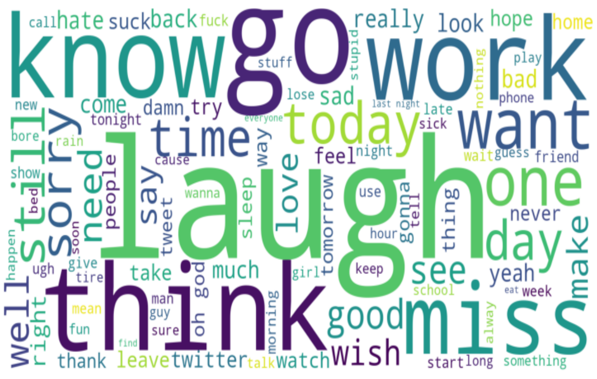

Fig.3: Word Cloud for negative sentiment

B. Preprocessing

Text preprocessing is an important step for Natural Language Processing (NLP) tasks. We need to pre-process the text to get it all in a consistent format. We need to clean our data into a digestible form. We have tried so many techniques to clean the texts in order to extract the essential part from each.

The first step of preprocessing pipelines is splitting documents into words. we performed tokenization for splitting each sentence into words faster. With the help of NLTK tokenize regexp() module [1], we were able to extract the tokens from string by using regular expression with RegexpTokenizer() method. The tokenization helps in interpreting the meaning of the text by analyzing the sequence of the words.The benefit of tokenization is that it gets the text into a format that is easier to convert to raw numbers.

After tokenizing, the text is cleaned by removing Some words were not good enough such as usernames, URLs, punctuation, hashtag, mention, special characters, slangs and numbers. Some extremely common words which would appear to be of little negative or positive value in helping select words. These words are called stopwords, then we eliminated them. All remain characters were converted to lower case enabling text to be normalized.

Next step in preprocessing is reducing inflected (or sometimes derived) words to their stem, base or root form, generally a written word form. Lemmatization is the process of converting a word to its base form. The difference between stemming and lemmatization is, lemmatization considers the context and converts the word to its meaningful base form, whereas stemming just removes the last few characters (i.e. books become book), often leading to incorrect meanings and spelling errors.Ultimately, the process of cleaning the text is completed.

The processed Data is divided into two datasets: training dataset 80% and test dataset is 20% of text size. The former dataset, development, was the model would be trained on, and the latter dataset, evaluation, upon which the model would be tested against. We used the TfidfVectorizer() method. The Tfidf, term frequency and inverse document frequency of the word across a set of documents to extract features from raw textual data and it is computed as it follows:

$$TFIDF = (t, d) \log \frac{|D|}{freq(t, D)}$$

Using this weighting schema avoids biasing the analysis by assigning too much weight to commonly used words within the topic covered in the reviews. In creating the Bag of Word, also unigram (one word in the same text) have been considered in the max_features of 10000. According to Fig.4

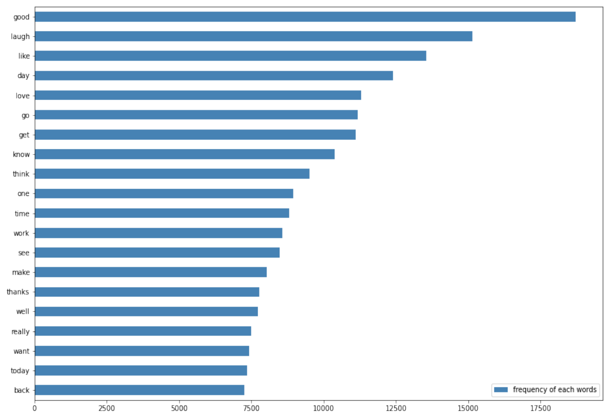

Fig.4:common words

Standardization of a dataset is a common requirement for many machine learning estimators. It scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one. So we used a StandardScaler modul from sklearn.preprocessing library to compute relevant statistics on the samples in the training set. Mean and standard deviation are then stored to be used on later data using transform.

C. Model selection

Having prepared our data, we have to select one model and tune and validate it among the following models:

- Logistic Regression: it is also one of the most useful analytic tools, because of its ability to transparently study the importance of individual features. logistic regression generally works better on larger documents or datasets and is a common default. LR models the probabilities for classification problems with binary data. It is an extension of the linear regression model for classification problems.
- Support vector machines (SVMs) are powerful yet flexible supervised machines learning methods used for classification, regression, and, outliers' detection. SVMs are very efficient in high dimensional spaces and generally are used in classification problems. SVMs are popular and memory efficient because they use a subset of training points in the decision function. The main goal of SVMs is to divide the datasets into number of classes in order to find a maximum marginal hyperplane (MMH) which can be done in the following two steps:
  - Support Vector Machines will first generate hyperplanes iteratively that separates the classes in the best way. After that it will choose the hyperplane that segregate the classes correctly.
  - After that it will choose the hyperplane that segregate the classes correctly
- Bernoulli Naive Bayes: it can work extremely well (sometimes even better than logistic regression) on very small datasets (Ng and Jordan,2002) or short documents (Wang and Manning, 2012). naive Bayes is easy to implement and very fast to train (there's no optimization step). So

it's still a reasonable approach to use in some situations. These classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers.

Models	**macro avg**
Logistic Regression	0.78
Support Vector Machines	0.77
Bernoulli Naive Bayes	0.76

TABLE I
COMPARISON OF MODELS

These three models have very similar performance but, looking at the f1-score (macro avg), Logistic Regression model seems to perform a little better than others.Table I.

D. Hyperparameters tuning

Regarding the Logistic Regression model, the following hyperparameters are adjusted:

solver: to use in the optimization problem [2]

penalty:control properties of the regression coefficients, beyond what the pure likelihood function (i.e. a measure of fit)[2]

class_weight: it basically means replicating the smaller class until we have as many samples as in the larger one, but in an implicit way.[2]

C: inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.[2]

fit_intercept: specifies if a constant (a.k.a. bias or intercept) should be added to the decision function.

A 80/20 Train-Test split has been performed and runned by using LR in order to get the best parameters for our model. In Table II it is possible to see parameters'values and Every model present in the Table 2 is tested, we tried every combination of hyperparameters related to the respective model and achieved the best F1 score obtained on Public Leaderboard and during local tests.(Table II)

III. RESULTS

The best configuration for Logistic Regression is reached with min_df = 5, max_df=0.8, sublinear_tf = True, use_idf = True, max_features=9000 for TFIDF vectorizer. we set solver ='liblinear', fit_intercept = True ,class_weight ='balanced', penalty ='l2 for LR model and achieved the higher score than other hyperparameters in Table II. This could create perfect or almost to perfect predictions for those records. However this is only a supposition, and the real cause of the higher values on the public set could be others. On the other side, the public result is one of the lowest in the competition, hence

Models	**Hyperparameters**	**Values**
Logistic Regression	solver	'liblinear' 'newton-cg', 'lbfgs' 'sag', 'saga'
	penalty	'none', 'l1', 'l2', 'elasticnet'
	C	100, 10, 1.0, 0.1, 0.01
Support Vector Machines	kernels	'linear', 'poly' 'rbf', 'sigmoid'
	C	100, 10, 1.0, 0.1, 0.01
Bernoulli Naive Bayes	-	-

TABLE II
COMPARISON OF MODELS

there is surely room for improvement We can clearly see that the Logistic Regression Model performs the best out of all the different models being tried. It achieves nearly 78% accuracy while classifying the sentiment of a tweet. Although it should also be noted that the SVM Model is the fastest to train and predict on. It also achieves 77% accuracy while classifying.

IV. DISCUSSION

In this project, we have discussed the sentiment classification problem using the tweet dataset. We have discussed the results in detail to infer useful insights. We can experiment with the hyperparameter to gain more information from this work. We have used only the tweet text to model the classification task. However, there are a few other columns that can be used for further exploration.

REFERENCES

[1] "Nltk." https://www.nltk.org/api/nltk.html.

[2] "Scikit-learn library" https:/www.scikit-learn.org