4/16/2018

Sentimental Analysis

Zeyad Zanaty	3320
Youssef Fares	3228
Mostafa Ali Mansour	3250

Sentimental analysis 

1) Data set reader 

a. We downloaded the data set locally and then directed the path to (../ac1Imdb).
b. Afterwards we discard any file/folder whose name isn't train or test.
c. We build an array of directories holding the 2 keywords(pos, neg) because we will use it in training and testing data.
d. We extract the id and label of the review according to a regex expression then we sort them according to a certain lambda expression.
e. At last we open the file containing the review itself and return the whole object "review".

2) Pre-processing 

a. The first step is to tokenize the review itself and convert it to array of strings.
b. Second step is to remove stop words according to a certain regex in the English vocabulary.
c. We implemented a diversity of pre-processing methods.
d. Porter stemming, which is the most popular stemming technique.
e. Lancaster Stemming which has more than 120 rules for getting stem words.
f. Snowball Stemming as an extra feature to support other languages for later projects.
g. Lemmatization also converts a word into its base form. However, the root word also called lemma, is present in dictionary. It is considerably slower than stemming because an additional step is performed to check if the lemma formed is present in dictionary.
h. In lemmatization we first train a model to get the "part of speech" content, whether the word was a noun, verb, adverb or adjective.

i. Then, we lemmatize it according to its part of speech.
j. We have support all these pre-processing methods.
k. We repeat the same steps on the testing data.

3) Vectorizing 

a. The code is highly modularized.
b. In the Main class, we instantiate a vectorizer object and pass within its constructor, they name of the vectorization method, the training data, and the parameters (if any).
c. In the Vectorizer class, the constructor has several conditional lambda expressions according to the type of vectorization acquired in the Main class.
d. A method (vectorize) extract the review itself from the ID and classification (good or bad) and puts it in an array.
e. The array calls the lambda expression (vectorize_call)
f. By convenience, the lambda matches the type and return a numpy array.
g. In Count and TFIDF types, the implementation is really abstract.
h. We faced a problem however in Word2Vec and FastText, because in Count and TFIDF, the result would be an array on the whole review, but on the other two word embedding methods, it will result a matrix of matrices representing each word, this will result into a “Setting an array element into a value” error.
i. We tried to have a 2D array having the maximum and minimum values but it had terrible results.
j. Instead, we calculate the minimum amount of words acquired in a review, and set the rows of all the vectors representing all the words to that number.
k. This has improved the accuracy.
l. The same protocol is followed to vectorize the test data.

4) Classification

a. We have implemented all 8 classifiers in the PDF.
b. As mentioned, the code is highly modularized.
c. The class (New classifier) take the name of the classifier and sets its data and label matrix and vector.
d. Then, the function tune is used, which contains a bucket of parameters used accordingly to the called classifier.
e. We have a product method to calculate every combination of the classifier with the tuning parameters.
f. Afterwards, it fit the data and labels then calculate the score relative to the new test data.
g. Finally, the maximum score with its combination is returned.
h. We found out that the highest accuracy could be required is 90.6% with Random Forest of 70 trees.
i. Below is the graph of accuracies relative to different combination of Pre-processing, vectorizing and classification.

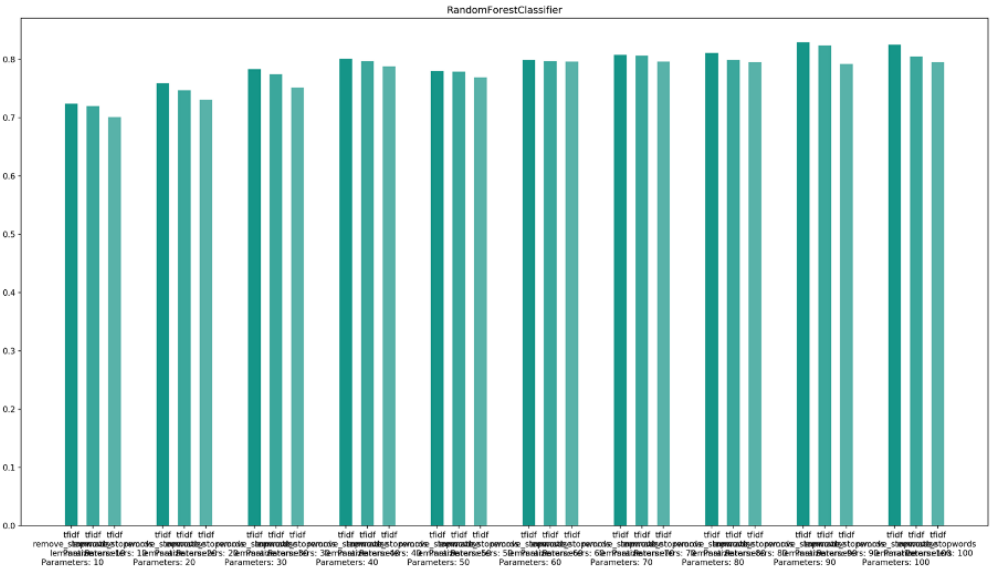

Figure 1 Random Forest with TFIDF and Lemmatization

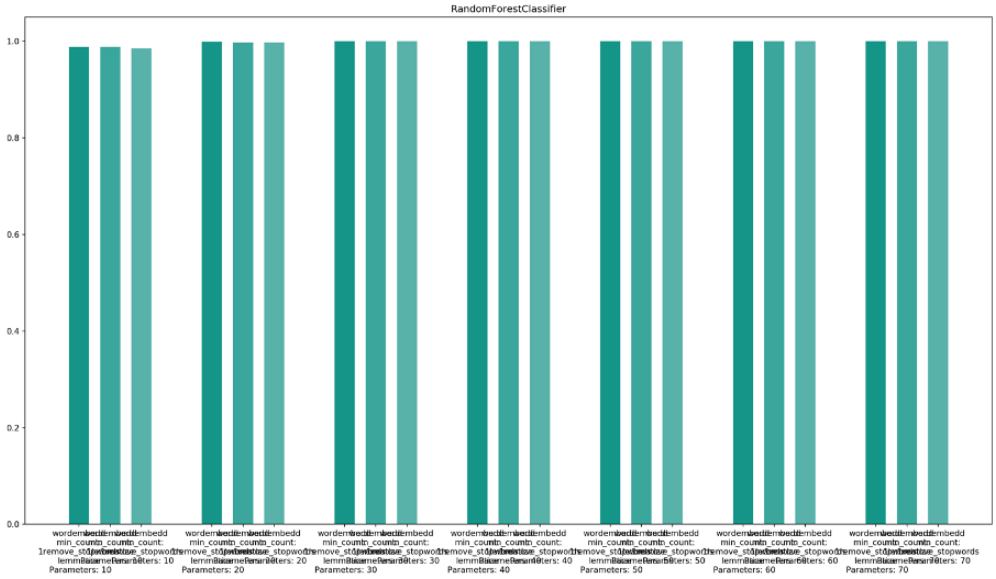

Figure 2 Random Forest with Word2Vec and Lemmatization

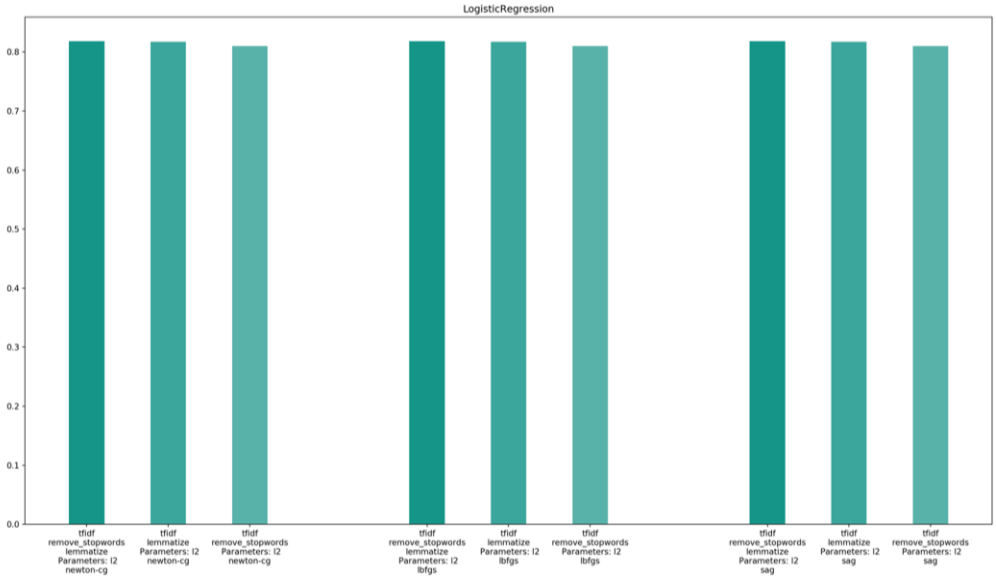

Figure 3 Logistic Regression with TFIDF

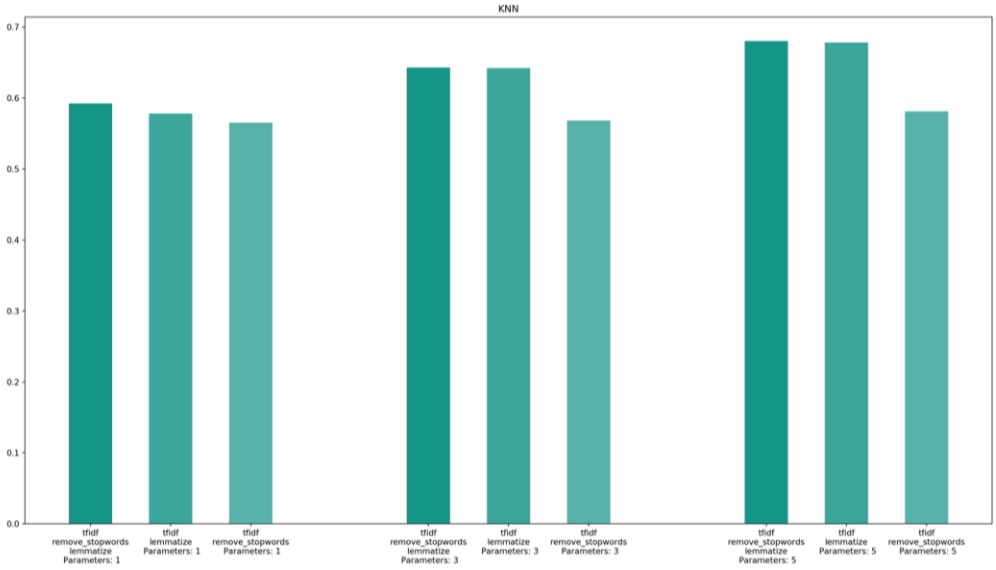

Figure 4 KNN with TFIDF

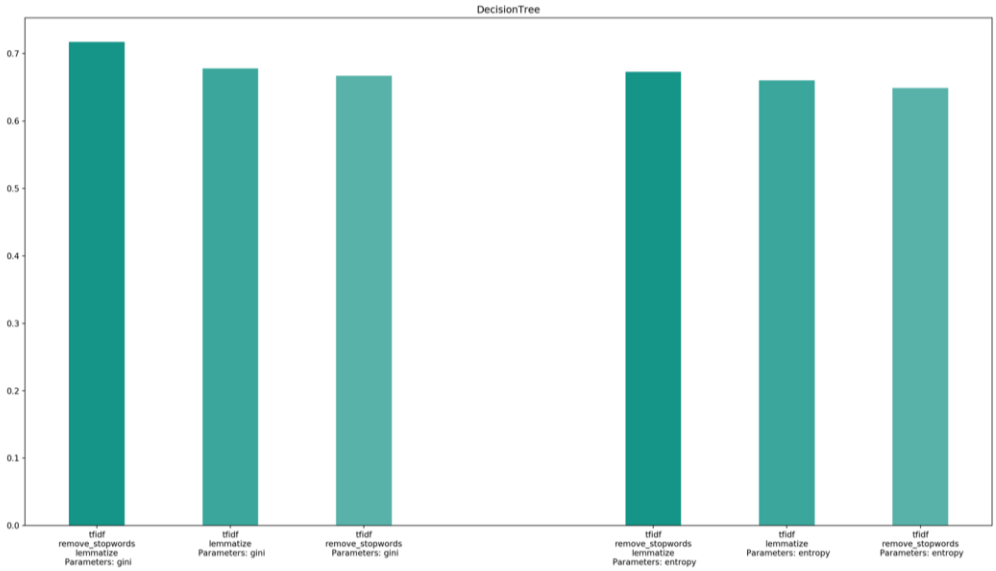

Figure 5 Decision Trees with TFIDF