Paris Diderot University M2 Data Mining Project

Twitter Writing Style Analysis

FAN Yi-Zhe | HSIEH Yung-Kun

Introduction

This project is to apply the machine learning techniques learned in the Master course "Fouille de Données et Aide à la Décision (Data Mining and Decision Support)" at Paris Diderot University. Our ideal final goal is to detect the plagiarism of text on the internet from intellectuals/copywriters. Within the limit of time (in 6 weeks of implementation), we decided to start with the twitter analysis in order to correctly recognize the tweets from different authors.

From Kaggle.com, we found 6 datasets of tweets of different celebrities as: Hillary Clinton (Politician), Donald Trump (President), Barack Obama (Ex-president), Adam Savage (Host of Myth Busters), Kim Kardashian (Actress), and Richard Dawkins (Ethologist). Our objective is to feed their datasets to our program and to predict the new input text of which celebrity-alike.

We implemented all our codes Python since there are many existing machine learning libraries in Python. The source code is on GitHub:

- https://github.com/QCTW/ParisVII_ProjectMachineLearning

Implementations

In our implementations, we have 2 user flows as in the Figure. 1. One flow is simply train/test our datasets with seven different classifiers. It will generate a pyplot bar-diagram to show the benchmarks of different classifiers. The other flow is an interactive mode which will start right after train/test and will prompt some message waiting your input.

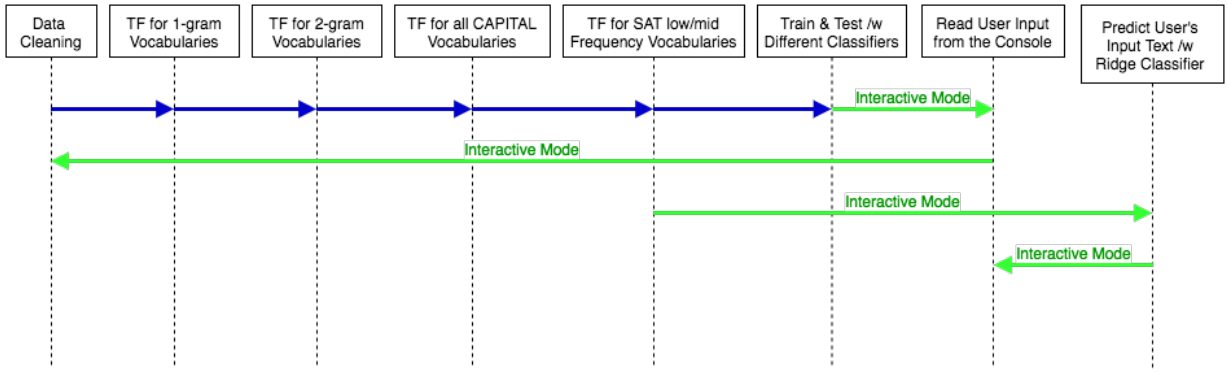


Figure 1: Program User Flow

Those steps are chained by the main.py. To run the program with simple train and test, type:

python3 main.py

To run the program with interactive mode (-i) to predict your input text, type:

python3 main.py –i

Raw Data Cleaning

To have some proper text for analysis, we pre-processed raw data files from Kaggle and marked it with different authors.

1. Use CSV library to read the raw data files.
2. For each line, replace the quoted(re-tweeted) text between two quotation mark with string "__QUOTE__". This is to make sure that the text is purely from the author.
3. For each line, replace the internet URL which begin with "https:\\" with string "__URL__".
4. For each line, replace the break-line ("\n") with blank string.

Features for Classifiers

To train our datasets, we use the following 3003 features:

1. Term frequency of 1500 1-gram vocabularies. We use `CountVectorizer` from sklearn library with parameters:
   - min_df equals to 2: to eliminate the appear only once IDs and low frequency vocabularies. We keep only words who appears in more than 2 documents.
   - max_df equals to (1/number of known classes): We considered that if a word appears once in each documents of certain class, such a word appears too frequently and should be eliminated.
   - max_features equals to 1500: We keep only the top 1500 frequent words for comparisons.
   - Use default stop words from sklearn to eliminate general high frequency words.

2. Term frequency of 1500 2-gram vocabularies.
   Same parameters settings as 1-gram vocabularies, but we want to have frequency counts with 2 consecutive words.

3. Term frequency of all capital vocabularies.
   We implemented all capital words count in features.py.

4. Term frequency of very difficult SAT vocabularies.
   We supposed that people who has higher education will write more "sophisticated", as the result in their text is that they tend to use difficult (rare) vocabularies more often. We use the vocabulary dictionary for preparing SAT (Scholastic Assessment Test) as a reference of vocabulary's difficulty.

   To make sure that words in different forms are treated and count properly. We use `PorterStemmer` from nltk library to transform both the words of datasets and the vocabularies in the SAT dictionary to its root form. For example, word "pythoning" will transform to "python".

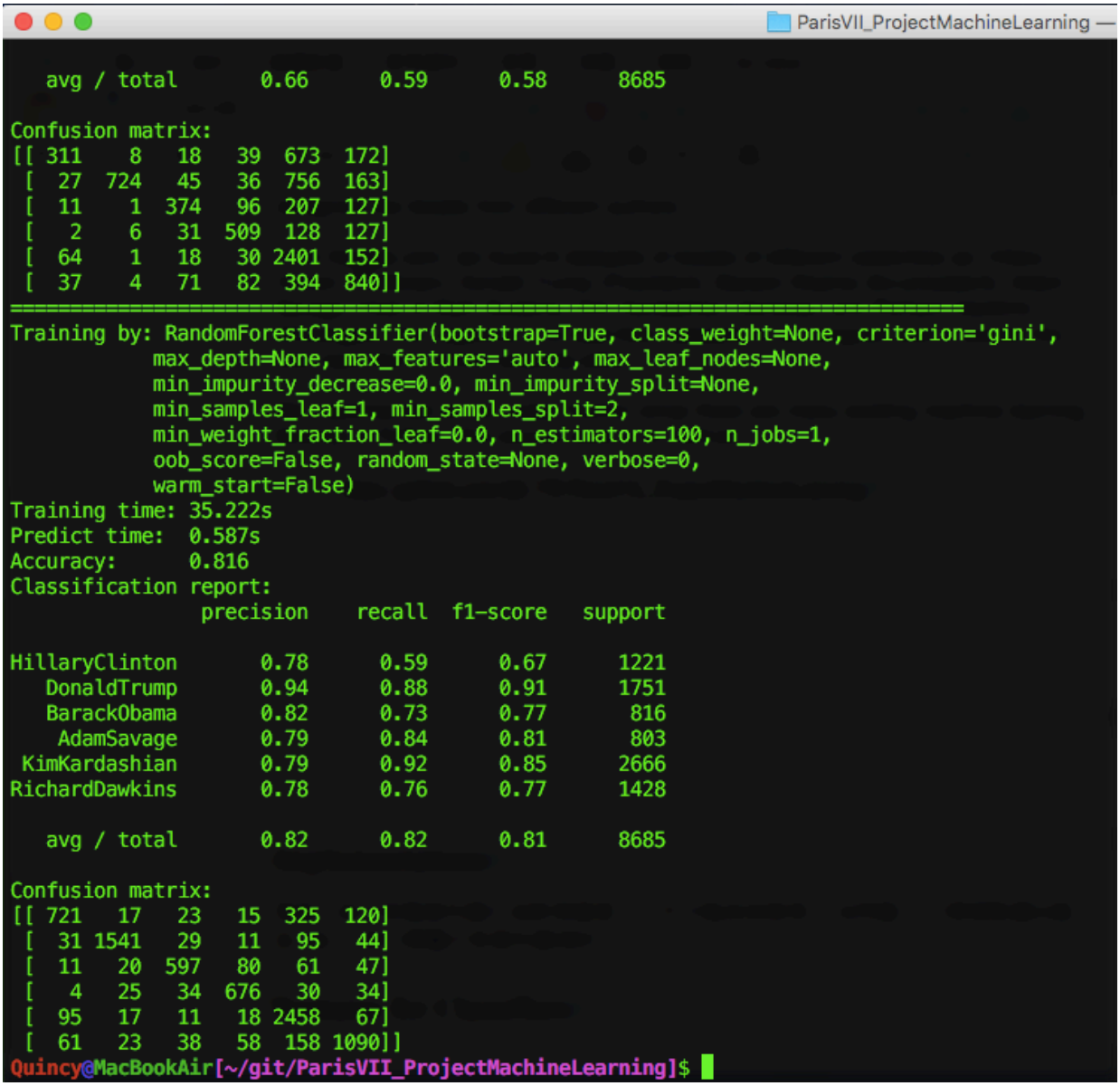
5. Term frequency of medium difficult SAT vocabularies.
   Same process as very difficult SAT vocabularies above.

Classifiers & Test

The classifiers and benchmark test are inspired by the sklearn tutorial for our training:

http://scikit-learn.org/stable/auto_examples/text/document_classification_20newsgroups.html#sphx-glr-auto-examples-text-document-classification-20newsgroups-py

Here is the result of our running classifiers and test.


ParisVII_ProjectMachineLearning —
avg / total 0.66 0.59 0.58 8685
Confusion matrix:
[[311 8 18 39 673 172]
[27 724 45 36 756 163]
[11 1 374 96 207 127]
[2 6 31 509 128 127]
[64 1 18 30 2401 152]
[37 4 71 82 394 840]]
Training by: RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
max_depth=None, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
oob_score=False, random_state=None, verbose=0,
warm_start=False)
Training time: 35.222s
Predict time: 0.587s
Accuracy: 0.816
Classification report:
precision recall f1-score support
HillaryClinton 0.78 0.59 0.67 1221
DonaldTrump 0.94 0.88 0.91 1751
BarackObama 0.82 0.73 0.77 816
AdamSavage 0.79 0.84 0.81 803
KimKardashian 0.79 0.92 0.85 2666
RichardDawkins 0.78 0.76 0.77 1428
avg / total 0.82 0.82 0.81 8685
Confusion matrix:
[[721 17 23 15 325 120]
[31 1541 29 11 95 44]
[11 20 597 80 61 47]
[4 25 34 676 30 34]
[95 17 11 18 2458 67]
[61 23 38 58 158 1090]]
Quincy@MacBookAir[~/git/ParisVII_ProjectMachineLearning]$

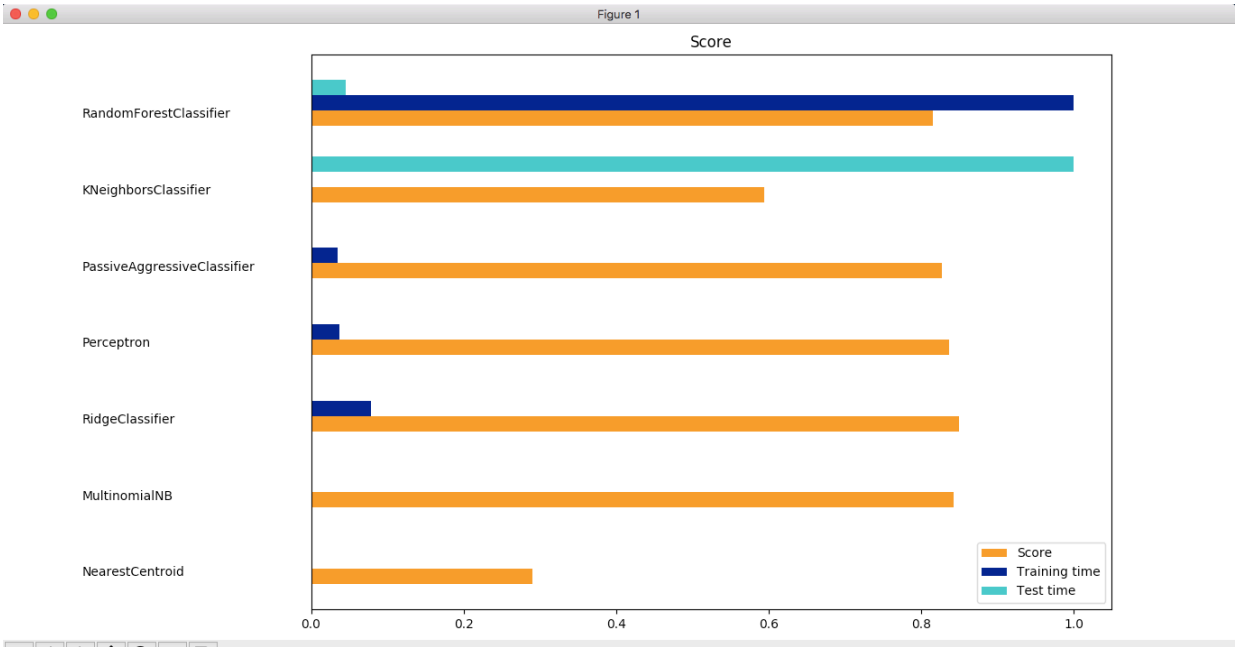


Figure 1
Score
RandomForestClassifier
KNeighborsClassifier
PassiveAggressiveClassifier
Perceptron
RidgeClassifier
MultinomialNB
NearestCentroid
Score
Training time
Test time
0.0
0.2
0.4
0.6
0.8
1.0

Difficulties

The problem that we had:

1. To predicate one tweet, we need to transform the features in this tweet to the features of our training data.
2. To add new feature to the sparse matrices, we need to merge two sparse matrices.
3. To find the best parameters for seven classifiers. In most of time, we just use the default one.
4. Cross validations find best k for the random forest classifier.
   Result of the validation cross:
   Divide the data set into 5 parts for validation cross...
   For k 50 Average_score is 0.803410591175
   For k 100 Average_score is 0.80605884285
   For k 150 Average_score is 0.806097127419
   For k 200 Average_score is 0.807172057208
   For k 250 Average_score is 0.806596153063
   For k 300 Average_score is 0.805789996208
   For k 350 Average_score is 0.806327527525
   For k 400 Average_score is 0.804791841771
   For k 450 Average_score is 0.806135750857