CS 410 Text Information Systems Fall 2018 Final Project Documentation

Group member: Jae Wook Lee(jlee759), Hye In Kim(hikim2)

Teaching Assistant: Kanika Narang

Professor: ChengXiang Zhai

Table of Content

I. Introduction

This is a final project for the course CS 410 - Text Information Systems that focuses on general computational techniques for managing and analyzing large amount of text data that can help users manage and make use of text data in all kinds of applications. Throughout the semester, we worked as a team of 2 to develop a Fraud Detection Program that would determine whether a review from a specific app was written with an intention to negatively affect the app.

This documentation includes the methodology behind the implementation, how it was implemented, how to use the software with an example, one use case of this software, and a conclusion.

II. Methodology

To accomplish our goal, we have analyzed dataset of mobile app reviews using the scikit-learn tool for text classification based on sentiment analysis. We followed the steps below for our project:

Step 1. App reviews collection

To evaluate on the machine learning algorithms for text classification, we have used the app reviews dataset found online to test our methods of reviews classification. It consists of 902 reviews with their labeled sentiment ('False', 'True', or 'Spam').

We have also used HTTP requests and json to get a review collection of 9 social apps with 2409 reviews, a review collection of 6 game apps with 1535 reviews, and a review collection of 6 education apps with 1677 reviews.

Step 2. Data preprocessing

For preprocessing, we uses unigram and removes the stopwords and punctuations from the data string in the collection.

Step 3. Feature Selection

Feature selection is a process to decide which subset of features are mostly related to increase the level of accuracy. In this project, we have used three feature methods used for classification task of sentiment analysis with stopwords methods and one clustering method.

Step 4. Sentiment Classification algorithms

In this step, we will use three sentiment classification algorithms and a clustering algorithm. The three sentiment classification algorithms used are SGD, SVM, and Naive Bayes. The one clustering algorithm we will try out is k-means clustering.

1) *Stochastic Gradient Descent* (SGD): The SGD classifier is simple yet efficient. It learns from the linear classifiers such as SVM and Logistic Regression.
2) *Support Vector Machine* (SVM): The SVM classifier is a linear classifier, meaning the classification decision is based on the value of the linear combinations of the data features. The goal of the SVM classifier is to find the optimal linear separators between classes.
3) *Naive Bayes* (NB): The NB classifier is the simplest and most widely used classifier based on the Bayes' theorem.

Step 5. Detection Processes

After the training process completes, we have to predict the output of the testing datasets. The possible results are listed below:

- True Positive (TP): Real reviews in testing data that are correctly classified by the model as Positive (P)

- False Positive (FP): Fake reviews in testing data that are incorrectly classified by the model as Positive (P)
- True Negative (TN): Real reviews in testing data that are correctly classified by the model as Negative (N)
- False Negative (FP): Fake reviews in testing data that are incorrectly classified by the model as Negative (N)

The

Step 6. Comparison of results

Using the results we have obtained from Step 5, we will be comparing the 3 classification algorithms by computing the following measures:

- Fake Positive Reviews Rate = FP/(FP+TN)
- Fake Negative Reviews Rate = FN/(TP+FN)
- Real Positive Reviews Rate = TP/(TP+FN)
- Real Negative Reviews Rate = TN/(TN+FP)
- Accuracy = (TP+TN)/(TP+TN+FN+FP)
- Precision = TP/(TP+FP)
- Recall = TP/(TP+FN)
- Fscore = (2*Precision*Recall)/(Precision+Recall)

III. Implementation

As the final project contains several .py files, we will go through all the files one by one.

1. fetch_review.py
    a. This makes an http request to apple itunes store to get all the recent reviews from the users so that we could test our implementation afterwards.
2. app_name_dict.py
    a. This file is to store constants. fetch_review.py uses this file to fetch reviews from apps that was specifically listed in this file. Feel free to add any more to the dictionary
3. analyze.py
    a. analyze(pred_data, label) function takes in the predicted label and the real label and computes true negative, false positive, false negative, true positive, precision, recall, fscore, and overall accuracy from the given labels.
    b. bargraph(...) generates a graph with all the data generated by analyze function for convenience
4. train.py
    a. parse_csv(file) parses the csv data file and stores necessary data in memory
    b. KMeansModel(features) function removes stopword, convert text to tf-idf, and utilizes sklearn.cluster.KMeans to process data.
    c. SVCModel(data) contains one inner class and one inner function to preprocess data. Then is utilizes sklearn.svm.SVC to process data.
    d. MultinomialBNModel(data) and SGDModel(data) both utilize CountVectorizer and TfidfTransformer to preprocess data. Then uses PipeLine to process preprocessed data utilizing sklearn.naive_bayes.MultinomialNB and sklearn.linear_model.SGDClassifier respectively.
5. test.py
    a. This function contains wrapper functions to run different kinds of classifiers.

IV. Use Case/Usage

Requires Python 3.7.0
First one needs label data for our classifier to learn what kind of sentiments corresponds to which. Please download training.csv from our github. Next, one should fetch app reviews to detect fraud reviews. Please utilize fetch_review.py and make any necessary changes to app_name_dict.py to add or remove app reviews from the data. Lastly modify test.py to run tests that you've added in app_name_dict.py.

```
-- command line
$ python3 fetch_review.py
….
….
….
$ python3 test.py
```

Use Case

Precondition:
The user has forked or download source code and labeled training data from Github

Main Flow:
The user is interested how much reviews in the app he is going to download is actually trustworthy.

Sub Flow:
1. The user adds the app id number and name to the app_name_dict.py.
2. Runs program that fetches all the review from app_name_dict
3. Test and analyze the output.