Twitter Sentiment Classification using sklearn

Brief

- **Due date and time: Wednesday, 11/23/16 at 11:59 pm**
- **Data: /data/train.csv and /data/test.csv**
- **Handin: following the handin requirement, the last section in this writeup**
- **Required files: README.md, [your-netid].py, REPORT.txt**

SETUP:
You will not require a separate setup step for this assignment as you would have already installed sklearn(scikit learn) for last assignment.

Overview

In this assignment, you will be performing sentiment analysis on tweets using scikit learn with different machine learning techniques. You will learn how to extract features from data and build supervised learning models to automatically classify tweets as either positive or negative (sentiment). Classifiers we will be examining are Naive Bayes, Logistic Regression, and Decision Tree.

Data

The /data directory contains two csv files: training.csv and test.csv.
It takes a reasonable amount of time to build a model. If in your case, it is too slow to use too much training instances, feel free to take a subset of the training data to do experiments. But you should know that, you cannot build a model with good performance on too little training data.
Data file format has 6 fields:

1. the polarity of the tweet (0 = negative sentiment, 1 = positive sentiment)
2. the id of the tweet (2087)
3. the date of the tweet (Sat May 16 23:58:44 UTC 2009)

4. the query. If there is no query, then this value is NO_QUERY.
5. the user that tweeted (robotickilldozr)
6. the text of the tweet

You will be only using the polarity and the text of the tweet. You should examine these data sets before starting the assignment.

Note:
The test set provided is only a subset of actual test set on which you will be evaluated. **You are not supposed to train on the test set**. So ensure that your model does not overfit.

Original data source: Sentiment140.

Supervised Classification

Supervised learning is a task of predicting a label (class) for the given test input, using the learning model trained on the labeled training inputs. For this assignment, you will train your model on training.csv and test it on test.csv. Both csv files contain sentiment labels (either positive or negative) for tweets. As shown in figure(in /data) during the training phase, the feature extractor is used to convert each input data to a feature set. Then, pairs of feature sets and labels are inputted into the machine learning algorithm to train a model. During the prediction phase, the same feature extractor is applied on the test data, and the extracted feature sets are inputted into the model to generate the predicted labels.

Feature Extraction

Bag-of-words model

The bag-of-words model, which is a vector-space representation of documents that represents each document (tweet) as a vector of words, disregarding word order.
Below is an example: Here are three documents, each document being one tweet.

i love you and you love me
you are my love
my love for you

Based on these three documents, a vocabulary list is constructed as:
'i'
'love'
'you'
'and'
'me'
'are'
'my'
'for'

which has 8 distinct words. And using the indices of the vocabulary list, each document is represented by an 8-entry vector:
1. [True, True, True, True, True, False, False, False]
2. [False, True, True, False, False, True, True, False]
3. [False, True, True, False, False, False, True, True]

where each entry of the vectors refers to the existence of the corresponding entry in the vocabulary list. The first entry is only 'True' for the first tweet since in the vocabulary, 'i' only exists in the first tweet. Usually, if we had a longer document, we would define each entry of vectors to refer to the count of the corresponding entry in the dictionary. For example, the first tweet would be vectorized as [1, 2, 2, 1, 1, 0, 0, 0].

Sklearn provides a convenient package for Feature Extraction. For sentiment analysis, commonly used feature set is terms and frequency bag of words feature model. The feature sets you extract contain unigram features as each vocabulary in the feature space (the vocabulary list) is one word. You can experiment with other ngrams such as bigram (two-word feature) or trigram (three-word feature). For example, bigram features in the above tweets are 'i love', 'love you', 'you and', 'and you', 'you love', 'love me', 'you are' … Your feature space can also be a combination of different ngrams. For instance, you can have both unigrams and bigrams in

your vocabulary list. For simplicity, it will be sufficient to keep only unigrams in your feature space, but you are welcome to experiment with other ngrams. You are welcome to. experiment with other feature extraction models like you did in homework 3.

Tweet Processing

Let's take look at two example tweets.
I am SO excitedddddd for uiccs491! #uiccs @Azure
http://uiccs.github.io/datascience

i am so excited @ApacheSpark

In the above example, the feature space will contain some duplicate information for some features. For example, both 'I' and 'i' will be in the feature space as well as the pair of 'so' and 'SO', and the pair of 'excited' and 'excitedddddd'. In addition, both the singular and the plural forms of a word will exist in the feature space. These duplicates cause the feature space to explode. It would be a good idea to group similar words as a single feature and keep meaningful words. Here are some suggestions to process tweets before building your vocabulary list and extracting features:

- Lowercase all characters
- Apply stemming using Porter Stemmer
- Strip punctuations
- Replace two or more occurrences of the same character with two occurrences. i.e. 'excitedddddd' to 'excitedd'
- Replace #xyz with xyz
- Replace a word contains www. or http(s):// with URL so that we can treat all the urls the same
- Replace a word contains @someuser with AT_USER so that we can treat all the users the same
- Ignore words that don't start with an alphabet
- Use the stop words list to filter out low value words such as 'the', 'is' and 'on'.
- … and many more

You are required to lowercase all characters, but you are free to experiment with other options. You probably want to use regular

expressions to implement some of these options. You should pick the processing options that are reasonable and give you the best performance.

Training

In this assignment, you will be experimenting with three classifiers: Naive Bayes, Logistic Regression, and Decision tree.

You will use classifer modules from Sklearn Supervised learning libraries.

Naive Bayes

Naive Bayes assumes that all features are independent from each other (in other words, each feature's weight is set independently), and uses the prior probability (i.e. what percentage of tweets are labeled as positive in the training set?) and the likelihood (i.e. what is the probability of seeing a word 'happy' in positive tweets?) to determine the posterior probability (i.e. if a tweet contains words 'happy', 'congratulations' and 'great', what is the probability of classifying this tweet as positive?) in order to classify the inputs. This article has a good example of Multinomial Naive Bayes, which uses word counts as its features. Since our documents are very short (140 characters limitation), we will be using Bernoulli Naive Bayes, which uses boolean values for each feature.
You will use Naive Bayes in sklearn, specifically Bernoulli Naive Bayes.

Logistic Regression

Logistic Regression essentially finds coefficients for the linear decision function. For example, it determines how much it will weigh a feature 'happy' when deciding the tweet is positive or negative. In contrast to Naive Bayes, whose feature weights are set independently, Logistic Regression sets all the weights together. Let's say there are two correlated features that are useful predictors. In this case, Naive Bayes will give both of them strong weights so those are double-counted, whereas Logistic Regression will compensate by weighting them lower.
You will use Logistic Regression in sklearn.

Decision Tree Learning

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. It is one of the predictive modelling approaches used in statistics, data mining and machine learning. Tree models where the target variable can take a finite set of values are called classification trees. In these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.
You will use Decision Trees in sklearn.

Validation

Once the model is trained, you need to see how it performs. You can evaluate on the training dataset to check if the model gives you reasonable performance (accuracy) on the data it was trained.

Let's say your classifier is performing well (i.e. ~80% accuracy on training set). It is now very tempting to stick with the current model and test it on the test set. However, it is possible that your model 'overfits' on training set. In other words, your model might perform well with known data (training set), but when it encounters a new data set (test set), it might not yield similar levels of high performance.

To avoid overfitting, it is a common practice to hold out parts of the training data as a validation set. You will use K-fold cross-validation to validate your training model. In K-fold cross-validation, the original sample is randomly partitioned into k equal size partitions. Of the k partitions, k - 1 partitions are used as a training set and the remaining partition is used as a validation set. This process is repeated over k folds to generate k results that are averaged to produce a single estimation. When experimenting with different models with different parameters and different features, you should cross-validate each model and pick the model that gives the best cross-validation performance.

You might find this link useful

Testing

Once you found the best model, let's first test on a single tweet. Here is one tweet:

It is possible to be 'madly in love' with someone, when it is really just an attraction to someone who can meet your needs.

To test on the test set, you can follow the same procedure to get the predicted label and the prediction probability.

To attain comprehensive statistics, you can use Evaluation Metrics in sklearn. Please follow the code snippet, use multiple evaluation metrics to analyze your model, such as precision, recall, F-measure, ROC etc.

Write up(Report)

For this project, you must write a project report , answering all questions below (label your answers 1. to 10., as unlabelled answers will not get a grade) elaborating on any additional work you did. You should state the problem you have solved, the setting, the procedure and the findings, as if you are writing a blog article or research paper. That is, your answers must be clear,complete, but concise (avoid repetitions).

1. Describe your tweet processing steps.
2. Describe your feature space. Did you decide to use unigrams, bigrams, or both? What is the size of your feature space?
3. Describe any extra work (i.e. parameter tuning) you did on three classifiers: NB, LOG and Decision Tree(DT). How did it help?
4. For three classifiers (NB, LOG and DT), report training accuracy, 10-fold cross-validation accuracy, test accuracy, avg precision/recall/f1-score on test, and the confusion matrix on test. Any findings?
5. For three classifiers (NB, LOG and DT), plot training accuracy, 10-fold cross-validation accuracy and test accuracy together using

matplotlib. You can check out this tutorial. Which classifier overfits the most?

6. Describe the following terms in the context of the assignment: precision, recall, f1-score, confusion matrix (true positive, true negative, false positive, false negative).
7. For NB and LOG, plot the ROC curve and report the area under the curve. What do you learn from the ROC curve?
8. Report top 20 most informative features for all three classifiers. Any findings?
9. Which classifier performs the best? Why?
10. Using the best classifer, print some test tweets that are classified correctly or incorreclty along with their prediction probabilities. Among correctly classified tweets, print 5 tweets with highest predicted probabilities. Repeat this for incorrectly classified tweets.

Handing in

For submitting to blackboard, create a folder with your NetID, put all the required files in the folder and zip it into [your NetID].zip, not other file extension.
The folder you hand in must contain the following:

- **README - text file containing anything about the project that you want to tell the TAs.**
- **[net-id].py - please put your all the code in this file. You should write detailed explanation/comment of each step about the methods and the reasons you do some operation. If there is some additional packages installed, please include the installation instruction in the first markdown cell in your notebook.**
- **REPORT.txt**

Late submission is not acceptable. The submission link in Blackboard will disappear immediately at the deadline. You could submit multiple times, only the latest version will be graded.

Cheating/Plagiarism Policy:

Any cheating may result in a **fail grade** in the course **and/or be reported to the university higher authorities for appropriate action**.