CMPS 143 - Assignment 4

Due Friday, April 29, 11:55 PM

1 Classification of Restaurant Reviews

1.1 Overview

In assignment 3, you developed a Naive Bayes classifier for classifying restaurant reviews as *positive* or *negative*. We will continue to work with the same data from user generated restaurant reviews from the site we8there[1]. The goal of this assignment is identical to that of Assignment 3: Given the text of a restaurant review, we'd like to know whether it is expressing a positive opinion about a restaurant or a negative opinion.

1.2 DecisionTree-Baseline Sentiment Classification

In this part of the assignment, you will use your previously extracted feature vectors to train a Decision Tree classifier model to automatically label unseen reviews as positive or negative, then compare the results with the Naive Bayes classifier. Read Chapter 6, Section 4: Decision Trees in the NLTK book. You **must** run the Decision Tree classifier on your feature vectors and generate the confusion matrix.

1. Write a program for training a classifier. The aim here is to see if you can get better results than your previous Naive Bayes classifier by simply using a new classifier with the exact same features you extracted in the Assignment 3.

   (a) Reuse the training instances and the features you created previously. (This is a list of tuples: the first element of the tuple is the feature vector of the document (i.e., the dictionary of feature/value pairs). The second element is the name of the category (i.e., *positive* or *negative*)).

   (b) Train a classifier with the training instances created in the previous step using the DecisionTreeClassifier as such:
   `classifier = nltk.classify.DecisionTreeClassifier.train(train_data, entropy_cutoff=0, support_cutoff=0)`

   (c) Save the classifier to disk using the following code snippet.

   ```
   import pickle
   f = open('dt-classifier.pickle', 'wb')
   pickle.dump(classifier, f)
   f.close()
   ```

2. Write a program that classifies reviews using your trained model. It should take at least three arguments:

   (a) The first should be the trained classifier model.

   (b) The second should be the file with the reviews in it.

   (c) The third should be the file your program writes its results to. You should report the accuracy of your classifier on the input dataset (*development* or *testing*) using the `nltk.classify.accuracy` method, and output that to this file. You should also apply the `nltk.classify.ConfusionMatrix(reference labels, predicted labels)` method that was demonstrated in class for the movie reviews in the `evaluate` method, and output that confusion matrix to this file.

[1] `https://www.we8there.com/`

Your program can take more that 3 arguments if needed. Please add comments in the beginning of your code to describe any additional arguments.

You can read in your trained classifier using the following code snippet.

```
import pickle
f = open('dt-classifier.pickle', 'rb')
classifier = pickle.load(f)
f.close()
```

3. Run your classification program on the development data. Save the output to `dt-output-dev.txt` (i.e. this filename should be the third argument you pass in to your program).

4. Run your classification program on the test data. Save the output to `dt-output-test.txt`.

1.2.1 Results

The responses to the following questions should be included in a file called `output-baseline.txt`. You will find the necessary information to answer these questions in various places; you should just copy and paste them into a single file.

1. Report the accuracy of your classifier on both the development and test data.

2. Report the confusion matrix generated from your new DecisionTree classifer, as well as your NaiveBayes classifer from the previous assignment, given the testing data. You do not need to report the confusion matrix for the development data in this file. (To get the confusion matrix for the Naive Bayes classifier, you can pass in to your program the `baseline-classifier.pickle` that you created in Assignment 3.)

1.3 Sentiment Classification Competition PART 2

For this next part of the assignment, **you must implement LIWC features**. In Assignment 3, you were asked to use the relative frequency of each feature. For this assignment, we want you to **use the value of each feature using "binning"**. There is an example of how to do this in `movie_reviews.py`.

Once you have made these two changes/additions to your code, play around with various combinations of features. For every combination you try, you should train it with both the Naive Bayes classifier and the Decision Tree classifier. Additionally, for the Naive Bayes classifier ONLY, you should perform feature selection to determine what number of features returns the highest accuracy. An example of how to do this was given to you in main in `movie_reviews.py`. It loops through the 10,000 best features returned by `most_informative_features`, and for each loop it looks at some subset of the features. For every loop, it returns the accuracy of that subset of features on the development data. Once it has determined which subset of features produces the highest accuracy, it uses those same features to evaluate the test data.

As noted before, for every combination of features you consider, you should train it on both the Naive Bayes classifier and the Decision Tree classifier. For DecisionTree, just report the accuracy. For NaiveBayes, once you have determined the number of features that returns the highest accuracy, you should report that accuracy and the number of features that produced it. You should create a table containing all this information, along with some indication of what the feature set included. An example results table can be seen in Table 1.

Report your results table in a file called `results.pdf` (or, if you can format it nicely so that it is easy to read in a file called `results.txt`). You are required to report at least 5 different combinations of features. Since the goal is to produce the most accurate classifier in the class, we suggest trying more.

Once you have found the combination of features that produces the highest accuracy, save your classifier model to a file named `restaurant-competition-model-P2.pickle` and submit a program named `restaurant-competition-P2.py` that classifies reviews in the same way as in the previous section. Again,

	Naive Bayes		Decision Tree
Feature Set	Accuracy	# of features	Accuracy
Unigrams	0.5	512	0.5
Bigrams	0.5	64	0.5
Unigrams + LIWC	0.5	8192	0.5
Unigrams + bigrams + POS-uni + POS-bi + LIWC	0.5	128	0.5

Table 1: Example Results Table

we will test your classifier on held out test data and report your classifier's accuracy along with everyone else in the class.

1.3.1 Questions

The responses to the following questions should be included in a file called `output-best.txt`.

1. Report the accuracy of your best classifier on both the development and test data.
2. Apply the `nltk.classify.ConfusionMatrix(reference labels, predicted labels)` to your best classifier given testing data and copy it to the `output-best.txt` file.

1.4 What To Turn In

The following files should be zipped together into a single file. Some files that we asked you to generate are not listed here because you do NOT need to turn them in.

- From DecisionTree-Baseline:
  - The .py file for classifying unseen data.
  - The pickle file containing the new DecisionTree classifier.
  - Your `output-baseline.txt` file.
- From Sentiment Classification Competition PART 2:
  - Your `restaurant-competition-P2.py` file for classifying unseen data.
  - Your `restaurant-competition-model-P2.pickle` file containing your best classifier.
  - Your `output-best.txt` file.
  - Your `results.pdf` or `results.txt` file containing the results of the various feature combinations you performed.