# CMPS 143 - Assignment 4

Due Sunday, 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 part of the 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. In the second part of this assignment you will be doing sentiment classification on a new dataset of Blogs.

### 1.2 DecisionTree-Baseline Sentiment Classification

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

1. Write a program for training three decision tree classifiers and save each through Pickle:

   - *dt-word_features-model-P2.pickle*
   - *dt-word_pos_features-model-P2.pickle*
   - *dt-word_pos_liwc_features-model-P2.pickle*

   The aim here is to see if you can get better results than your previous Naive Bayes classifiers by simply using a new classifier with the exact same feature sets you extracted in the Assignment 3.

   (a) Reuse the training instances and the feature sets 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 three classifiers (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 classifiers to disk using Pickle (same as the last assignment).

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

   (a) The first should be one the trained classifier models

   (b) The second should be the file with the reviews in it (*development* or *testing*).

   (c) The third should be the file your program writes its results to. You should report the accuracy of your classifiers 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.

3. Run your classification program on the development data. Save the output to `dt-FEATURE_SET-output-dev.txt`. 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-word_features-output-test.txt* and *dt-word_pos_features-output-test.txt*

### 1.2.1 Results

The responses to the following questions should be included in a file called `output-comparison.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 both of your Decision Tree classifier models and Naive Bayes classifier models on both the development and test data.

2. Report the confusion matrix generated from your new DecisionTree classifers, as well as your Naive-Bayes classifers from the previous assignment, given the testing data. You do not need to report the confusion matrix for the development data in this file.

## 1.3 New Features: Binning and Word embeddings

**Binning:** 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`.

**Word Embeddings:** Create a feature set by embedding the review text using pre-trained word embeddings. We will be using the Word-2-Vec pre-trained word embeddings that can be downloaded[2] from Google. We have provided you with a stub code file `word2vec_extractor.py` that implements some helper functions useful for turning words and sentences into features. You will need to install Gensim[3] in order to run the word-2-vec stub code.

## 1.4 Run Experiments

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 is 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.

[2] `https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit`
[3] `/radimrehurek.com/gensim/install`

| | **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

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

#### 1.4.1 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.

## 2 Classification of Blogs

In this part of the assignment you will be doing sentiment classification on a new dataset of Blogs. Also, we will be experimenting with a new Machine Learning model called Support Vector Machine.

In each assignment so far we have been using the machine learning models available in NLTK. For this part of the assignment we will be using a different Machine Learning library for Python called scikit-learn[4]. Instructions for installing scikit-learn can be found at `scikit-learn.org/stable/install` .

### 2.1 The Blogs Dataset

You will be using a different dataset for this part of the assignment. The dataset is comprised of Blog entries that each describe a positive or negative event, similar to the ones you wrote as part of your first assignment. The training set `stories-train` has 1,000 Blogs — 500 positive and 500 negative — and the development set `stories-development` has 200 Blogs. We also have two test sets, `stories-test1` contains 60 Blogs that the class wrote for assignment 1. And `stories-test2` contains 160 Blogs that mix blogs come from the same Blog dataset with class written blogs. You will also evaluate your trained classifiers on these two test dataset.

### 2.2 Classification

First, write a program that uses the scikit-learn versions of Naive Bayes and Decision Tree classifers to classify the sentiment of the Blogs:

1. Build feature representations for each of the Blogs in the dataset. You should use the same types of features that you used on the restaurant reviews.

[4]http://scikit-learn.org/

2. Train classifiers using the scikit-learn versions of Naive Bayes and Decision Tree classifers. NLTK provides a wrapper around the scikit-learn classification models so that you do not need to change the format of feature vectors. **See the lecture slides for an example of using scikit-learn classifiers with the NLTK wrapper.**
3. evaluate your classifiers on both the development data and test data (two sets). Save the results in the results.pdf file, the same way you did in the previous section.

Now, we will use a new type of Machine Learning model called the Support Vector Machine (SVM) to classify the blogs. Write a program that:

1. Train an SVM using the word_features, word_pos_features, and word_pos_liwc_features feature sets. Report the classifier's accuracy on the development data and test data.
2. Train an SVM using only the word embedding features and report its accuracy on the development data and test data.

### 2.3 Results

Create a table of result for the Blogs data similar to Table 1 and include it in your `results.pdf` file. Include all results for the the scikit-learn versions of Naive Bayes and Decision Tree classifers, and the SVM classifiers in the results table.

## 3 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 classifiers:
  - The .py file for classifying unseen data.
  - The pickle files containing the new DecisionTree classifiers.
  - Your `output-comparison.txt` file.
- From the Binning and Word Embeddings in section 1.3:
  - 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.
- From the Blogs classification:
  - The Python programs that you wrote.
  - A table of classification results in the same `results.pdf` file mentioned above.