CMPS 143 - Assignment 4

Due Monday, Feburary 12, 11:59 PM

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.

1.1 NLTK Classifiers

First, we want to compare your feature vectors from assignment 3 which use binning against a new model which just uses the raw count of the unigrams and bigrams. For example, imagine the word *the* occurs 10 times, *food* occurs 100 times, and the bigram *the food* occurs 1 time. You should construct your string vectors the same as before except now using the raw counts instead of the binned value. Hence your new feature vector from the previous example would now be:

```
UNI_the:10
UNI_food:100
BIGRAM_the_food:1
```

1. Write a program that classifies reviews using your trained models. It should take at least two arguments (Your program can take more than 2 arguments if needed. Please include a README with the command that runs your program) :
   (a) The first should be the file with the reviews in it (*restaurant-development.data* or *restaurant-testing.data*).
   (b) The second should be the file your program writes its results to. Your program should write the accuracy of your classifiers on the input dataset (*development* or *testing*) using the `nltk.classify.accuracy` method. And your program also writes to this file the result of `nltk.classify.ConfusionMatrix(reference labels, predicted labels)` method. Note that you don't need to turn in this output file however you will copy paste these results in a single file named `all-results.txt`.
2. Record accuracy of your classifier on *development* and *testing* datasets in a result table as shown in Table 1 and save in `all-tables.pdf`.

1.1.1 Naive Bayes Feature Selection

Using the model which performed best in the previous step, we now want to do **feature selection** to determine what number of features return the highest accuracy. In this assignment, **we are only interested in using the WORD based features, and can disregard the LIWC features.** An example of how to do feature selection is given to you in the main of `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 *testing* data. **You need to retain the single best set of features which are selected here as you will use them throughout the rest of the assignment.**
Retrain your Naive Bayes classifier only using this best set of features to compare it against your previous iterations.

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

1.2 Run Experiments

Once you have made these changes/additions to your code, play around with various combinations of features.

1. Run your best classifier on the test set. Highlight the cell in Table 1 that corresponds to that classifier.
2. 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 record all these information in a file called `all-tables.pdf`, along with some indication of what the feature set included. You are required to report at least 5 different combinations of features and you need to implement all additional changes explained in the previous section. An example results table can be seen in Table 1.

Feature Set	# of features selected	Model	Accuracy	Dataset
word_features	64	NB	0.5	dev
word_bin	512	NB	0.5	test
...	...	...	...	...

Table 1: Example of restaurant results table in `all-tables.pdf`

2 Scikit-Learn Classifiers

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[2]. Instructions for installing scikit-learn can be found at `scikit-learn.org/stable/install` .

2.1 Naive Bayes and Decision Tree Classifiers

First, write a program that uses the scikit-learn versions of Naive Bayes (BernoulliNB is equivalent to nltk version) and Decision Tree classifiers to classify the sentiment of the restaurant reviews:

1. Use the single best set of features you extracted earlier in this assignment.
2. Train classifiers using the scikit-learn versions of Naive Bayes (BernoulliNB) and Decision Tree classifiers. NLTK provides a wrapper around the scikit-learn classification models so that you do not need to change the format of feature vectors.
3. Evaluate your classifiers on both the development data and the testing data sets. Report the results in the `all-tables.pdf`. In your results table you will directly compare your Sci-kit learn Naive bayes with the NLTK Naive bayes for performance. An example is shown in Table 2.
4. Report accuracy and confusion matrix using the function described in Section **??** for each experiment you run in `all-results.txt`.

2.2 Support Vector Machine

Now, we will use a new type of Machine Learning model called the Support Vector Machine (SVM) to classify the reviews. We will also explore word embeddings and see how they compare against our best feature set.

2.2.1 Word Embeddings

Create a feature set by embedding the review text using pre-trained word embeddings. We will be using a custom version of Word2Vec pre-trained word embeddings. We have provided you with a stub code file `word2vec_extractor.py` that implements some helper functions useful for turning words, sentences and documents into vectors.

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

- You will need to download the provided embedding from Canvas.
- You will need to install Gensim[3]
- Put the model in the same directory of `word2vec_extractor.py` or you need to define the path in `word2vec_extractor.py` at the beginning `w2v = os.path.join("YOUR_PATH_TO_MODEL")`

2.2.2 Classification

Now we will train and evaluate our two SVMs.

1. Train an SVM using the single best feature set extracted earlier in the assignment. Report the classifier's accuracy on the development data and the test data.
2. Train an SVM using only the word embedding features and report its accuracy on the development data and the test data.

2.3 Results

Report all results for the the scikit-learn versions of Naive Bayes (BernoulliNB), Decision Tree and SVM classifiers in the restaurant results table similar to Table 2 and save it in `all-tables.pdf`. Note you can also try as many as combination of features and report in the table.

Feature Set	Model	Dataset	Accuracy
word_best_features	NB NLTK	test	0.5
word_best_features	NB NLTK	dev	0.5
word_best_features	NB Scikit Learn	test	0.5
word_best_features	NB Scikit Learn	dev	0.5
word_best_features	DT Scikit Learn	test	0.5
word_best_features	DT Scikit Learn	dev	0.5
word_best_features	SVM	test	0.5
word_embedding	SVM	test	0.5
...	...	...	...

Table 2: Example of restaurant result table in `all-tables.pdf`

2.4 Restaurant competition model

Once you have found the combination of features that produces the highest accuracy, write and submit a program named `restaurant-competition-P2.py` that classifies reviews using your trained models. It should take two arguments:

1. The first should be the directory path with the reviews in it. Hint The folder contains two sub-folders POS reviews and NEG reviews, and these sub-folders contain all the reviews that belong to that category in.txt format. So don't hard code the name of individual text files. You can get all the text filenames using os.listdir(PATH).
2. The second should be the file your program writes its results to. Your program should write the accuracy and confusion matrix to this file.

A default command for graders to test will look like this,

```
python3 restaurant-competition-P2.py
review-heldout result.txt
```

[3] `/radimrehurek.com/gensim/install`

Again include in the README if you need additional arguments. We will test your classifier on held out test data and report your classifier's accuracy along with everyone else's in the class (**Bonus points for the winners**).

3 What To Turn In

In total you need 2 prediction.txt files, 2 result reports, and all source codes (also include `word2vec_extractor.py`) if your program uses them. All 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 the NLTK Classifiers 1.2
  - The .py file classifying unseen data.
  - One prediction file, `nb-FEATURE_SET_PREDICTIONS.TXT`. Where FEATURE_SET is at your choice corresonding to the highlight cells.
- From the Scikit-Learn Classifiers 2.2
  - The .py programs you wrote to train and evaluate model. (It could be the same as restaurant-competition-P2.py if you design well)
- From Competition in Section 2.4
  - Your `restaurant-competition-P2.py` file for classifying and evaluating unseen data.
  - Your `restaurant-competition-model-P2-predictions.txt` file containing the predictions of your best classifier.
- All experiment results should be contained in two files:
  - `all-results.txt` contains accuracy and confusion matrix of each experiments you tried. Seperate by reasonable wording (or row number of the tables) to identify individual experiment.
  - `all-tables.pdf` contains two result tables as shown in Table 1 and Table 2.
- A `README` includes command to run each program and all necessary instruction.

What NOT to turn in. You don't need to turn in training, developement, testing data files as well as the word2vec model.