CMPS 143 - Assignment 4

Due Sunday, October 3, 11:59 PM

1 Overview

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

1.1 NLTK Classifiers

1.1.1 Binning

In Assignment 3, you were asked to use the relative frequency as value for word and part-of-speech (POS) features. For this assignment we want you to explore other values. First use the raw counts as value of each feature. Then use the binned value of raw counts. For example, imagine the word *the* occurs 10 times, *food* occurs 2 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. Hence your new feature vector from the previous example would now be:

`UNI_the:10 UNI_food:2 BIGRAM_the_food:1`

If you apply binning as shown in the code snippet below, your new feature vector from the previous example now becomes:

`UNI_the:3 UNI_food:2 BIGRAM_the_food:1`

```
def binning(count):
    """
    Results in bins of  0, 1, 2, 3 >=
    :param count:
    :return:
    """
    # Just a wild guess on the cutoff
    # you can experiment with different bin size
    return count if count < 2 else 3
```

Experiment with different value settings and binning cutoff, and determine which performed the best.

1.1.2 Naive Bayes Feature Selection

Using the model which performed best in the previous step, we now want to perform **feature selection** to determine what number of features return the highest accuracy. In this assignment, **we are only interested in using the word_features (unigram bigram and trigram), and can disregard the POS, LIWC, opinion 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 experiments.

1.2 Procedures

1. Unzip asg4-data.zip and put it in the same directory as other .py files. Keep the file structure unchanged in this folder.
2. Write/Modify imdb-competition-P2.py that classifies reviews using your trained models. It should take arguments (Your program can take more arguments than those in the stub code. Please include a README with the command that runs your program).

   Below is an example of command to run the stub code to train a Naive Bayes classifier and evaluate on development data set,

   ```
   $ python3 imdb-competition-P2.py -isTrain -cls nb \
   -train imdb-training.data -eval imdb-development.data \
   -c nb-word_features-classifier.pickle -o nb-word_features-dev.txt  \
   -f word_features
   ```

3. Your program should write the accuracy of your classifiers on the input dataset (`imdb-development.data` or `imdb-testing.data`) 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`.
4. 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`.

1.3 Run Experiments

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

1. Train and evaluate your model on development set and record results as shown in Table 1.
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.
3. Determine the best feature set you are going to use and run your best classifier on the test set.

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

Table 1: Example of movie reviews 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[1]. Instructions for installing scikit-learn can be found at `scikit-learn.org/stable/install` .

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

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 IMDB movie 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 1.2 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.

- You will use provided Glove embedding under asg4-data/glove-w2v.txt.
- You will need to install Gensim[2]

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 results on the development data and the test data in `all-tables.pdf` and `all-results.txt`.
2. Train an SVM using only the word embedding features and report its accuracy on the development data and the test data in `all-tables.pdf` and `all-results.txt`.

2.3 Neural Network

We will use the Multi-layer Perceptron (MLP) classifier [3] to classify the reviews.

1. Train the MLP classifier using only the word embedding features.
2. Report its accuracy on the development and test data in `all-tables.pdf` and `all-results.txt`.
3. Feel free to experiment more for your competition model

[2]`/radimrehurek.com/gensim/install`
[3]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

2.4 Results

Report all results on BOTH development and test data for the the scikit-learn versions of Naive Bayes (BernoulliNB), Decision Tree, SVM, MLP classifiers in the movie reviews 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 movie reviews result table in `all-tables.pdf`

2.5 IMDB competition P2 model

1. Determine the combination of features (name it as best_features in the code) and the classifier that produce the highest accuracy.
2. Train your best classifier and save it as `imdb-competition-model-p2.pickle`.

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

```
$ python3 imdb-competition-P2.py -eval imdb-heldout.data \
-c imdb-competition-model-p2.pickle -o result.txt -f best_features
```

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 one model named `imdb-competition-model-p2.pickle`, two XX_predictions.txt files, two 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 1.3
  - The .py file classifies unseen data with binning and feature selection implemented.
  - One prediction file that contains all the predicated labels on `imdb-testing.data`. Name it `nb-FEATURE_SET_predictions.txt`. Where FEATURE_SET is at your choice.
- From the Scikit-Learn Classifiers 2
  - The .py programs you wrote to train and evaluate model with various classifiers. (It could be the same as imdb-competition-P2.py if you design well)
- From Competition in Section 2.5
  - Your `imdb-competition-P2.py` file for classifying and evaluating unseen data.

  - Your best classifier `imdb-competition-model-p2.pickle`.
  - Your `imdb-competition-P2-predictions.txt` file containing the predictions of your best classifier on `imdb-testing.data`.

- 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 asg4-data which has training, developement, testing data as well as the glove-w2v.txt and liwc data files.