Supervised Learning Techniques for Sentiment Analytics

Estimated time: 8 hours

In this project, you will perform sentiment analysis over IMDB movie reviews and Twitter data. Your goal will be to classify tweets or movie reviews as either positive or negative. Towards this end, you will be given labeled training to build the model and labeled testing data to evaluate the model. For classification, you will experiment with logistic regression as well as a Naive Bayes classifier from python's well-regarded machine learning package scikit-learn. As a point of reference, Stanfords Recursive Neural Network code produced an accuracy of 51.1% on the IMDB dataset and 59.4% on the Twitter data.

A major part of this project is the task of generating feature vectors for use in these classifiers. You will explore two methods: (1) A more traditional NLP technique where the features are simply "important" words and the feature vectors are simple binary vectors and (2) the Doc2Vec technique where document vectors are learned via artificial neural networks (a summary can be found here).

Submission Instructions:
Make your changes directly to the `sentiment.py` file and submit this file on moodle.

Project Setup

The python packages that you will need for this project are `scikit-learn`, `nltk`, and `gensim`. To install these, simply use the pip installer `sudo pip install X` or, if you are using Anaconda, `conda install X`, where X is the package name.

Datasets

The IMDB reviews and tweets can be found in the data folder. These have already been divided into train and test sets.

- The IMDB dataset, originally found here, that contains 50,000 reviews split evenly into 25k train and 25k test sets. Overall, there are 25k pos and 25k neg reviews. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Thus reviews with more neutral ratings are not included in the train/test sets.
- The Twitter Dataset, taken from here, contains 900,000 classified tweets split into 750k train and 150k test sets. The overall distribution of labels is balanced (450k pos and 450k neg).

Project Requirements

You will be provided a basic stub for the project in `sentiment.py`. You job is to complete the missing parts of the file, look for the tag `# YOUR CODE GOES HERE` and fill in the missing parts to complete the code. Specifically, you must complete the following functions:

- **feature_vecs_NLP:** The comments in the code should provide enough instruction. Just keep in mind that a word should be counted at most once per tweet/review even if the word has occurred multiple times in that tweet/review.
- **build_models_NLP:** Refer to the documentation linked above for details on how to call the functions.
- **feature_vecs_DOC:** Some documentation for the doc2vec package can be found here. The first thing you will want to do is make a list of LabeledSentence objects from the word lists. These objects consist of a list of words and a list containing a single string label. You will want to use a different label for the train/test and pos/neg sets. For example, we used `TRAIN_POS_i`, `TRAIN_NEG_i`, `TEST_POS_i`, and `TEST_NEG_i`, where i is the line number. This blog may be a helpful reference.
- **build_models_DOC:** Similar to the other function.
- **evaluate_model:** Here you will have to calculate the true positives, false positives, true negatives, false negatives, and accuracy.

You should probably test on the IMDB data first, as this runs faster, particularly when using the doc2vec technique. Your outputs should be similar to the outputs shown below.

	output	
command	**Naive Bayes**	**Logistic Regression**
python sentiment.py data/imdb/ 0	predicted: pos neg actual: pos 10832 1668 neg 2374 10126 accuracy: 0.838320	predicted: pos neg actual: pos 10759 1741 neg 2057 10443 accuracy: 0.848080
python sentiment.py data/imdb/ 1	predicted: pos neg actual: pos 4739 7761 neg 2073 10427 accuracy: 0.606640	predicted: pos neg actual: pos 10362 2138 neg 1989 10511 accuracy: 0.834920
python sentiment.py data/twitter/ 0	predicted: pos neg actual: pos 67441 7559 neg 52250 22750 accuracy: 0.601273	predicted: pos neg actual: pos 67701 7299 neg 52239 22761 accuracy: 0.603080
python sentiment.py data/twitter/ 1	predicted: pos neg actual: pos 58686 16314 neg 50316 24684 accuracy: 0.555800	predicted: pos neg actual: pos 54009 20991 neg 33657 41343 accuracy: 0.635680