Supervised Learning Techniques for Sentiment Analytics

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.

A good overview of sentiment analysis and motivation of the Word2Vec and Doc2Vec techniques can be found here (for this project, you don't need to read or understand the two Mikolov papers that are mentioned). The code examples included in this link may be a useful reference for you as you work on the project, though it is not necessary to go through them; the skeleton code is setup in a way that will guide you through the project. However, you are welcome to to to improve upon the skeleton code if you would like.

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. There are also some large, pre-trained Word2Vec models in the public domain, that could be incorporated into training the Doc2Vec model, see the optional section below.

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.

The `gensim_models` folder contains the modified gensim source code that must be used if completing the optional section. This must be in the same directory as sentiment.py for your package import. This folder contains the source code for word2vec and doc2vec, you do not need to view or understand this source code; you are to simply work with `sentiment.py`.

Note that the modified source code no longer uses cython, which makes it significantly slower. If you don't plan on doing the optional section or find the project is taking too long to run, you should use the native version of gensim, which will be much faster. You can do this by commenting out the three imports that are there and instead using only:

```
from gensim.models.doc2vec import LabeledSentence, Doc2Vec
```

If you are using the native version of gensim, the vector embeddings from a model can be obtained via the command `model.docvecs['TRAIN_POS_452']`, whereas our modified gensim, and older versions, access the model directly, like `model['TRAIN_POS_452']`.

Finally, you should make sure you are using version 0.12.1 of gensim. With this version you also must have scipy version 0.15.1 in order for Cython to work properly. With pip you can obtain these versions via the following commands:

```
$ pip uninstall gensim
$ pip uninstall scipy
$ pip install --no-cache-dir scipy==0.15.1
$ pip install --no-cache-dir gensim==0.12.1
```

Datasets

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

- 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 (originally found here), contains 900,000 classified tweets split into 750k train and 150k test sets. The 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`. Your job is to complete the missing parts of the file, look for the tag `# YOUR CODE 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 to see how to call these functions for logistic regression and naive bayes.
- **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 (This blog may be a helpful reference for doing so). 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.
- **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/ nlp	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/ d2v	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/ nlp	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/ d2v	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

OPTIONAL: Including Pre-trained W2V model

There are some word2vec models available in the public domain that have been pre-trained on public corpora. However, incorporating this pre-trained model requires manually modifying the underlying gensim code. We have already completed this step and created the gensim_models.doc2vec_modified package.

If you would like to try incorporating a pre-trained model you must complete the code for the "_DOC_W2V" methods in `sentiment.py`. You will also need to download the pre-trained model (several gigabytes) from this website: https://github.com/idio/wiki2vec. Modify the path_to_pretrained_w2v variable in the code accordingly.

Note that including this pre-trained w2v model will require 20-30GB of memory and possibly several hours of training time.