CSCI 6350 (Barbosa)
Project 4: Sentiment Analysis using Multinomial Logistic Regression

Due: **See course calendar**. May not be turned in late.

Assignment ID: **proj4**
File(s) to be submitted: **proj4.ipynb**

Objective(s): Implement a Python program to determine multi-level sentiment (1 – 5 stars) from product reviews.

Project description: This project must run without modification on the NLP system – use a flat (1-level) directory structure to store files. Write a Python program that uses uses the polarity and intensity of words to assign one of five ratings to product reviews via a multinomial logistic regression classifier.

Requirements:

1. (**65%**) Your program must provide the functionality listed below, in a single file named ***proj4.py***:

- *Description* –You may use a sentiment/opinion lexicon to assign ratings, but it must be the one bundled with NLTK. The corpus containing the positive and negative words are in separate files located on the NLP system at *./nltk_data/corpora/opinion_lexicon/*
- Preprocessing – The reviews are in a file named ***reviews.txt***. Each review is on a single line that contains the review text, and its star rating (last integer on each line). The rating levels are:
    - 5 - Very Positive
    - 4 - Positive
    - 3 - Neutral
    - 2 - Negative
    - 1 - Very negative

  Your code should not make any assumptions about the number of reviews in the file, as it will be tested with different files. Choose an appropriate ratio to split this file into the files listed below for training and testing.

- *Input* – The input file is to be named ***reviews-train.txt***. Your program should read these reviews and process each, in turn.
- *Processing* – You may use the presence, number, and intensity of words (using part-of-speech, for example), and small n-grams (no larger than 4-grams) as features for each review. You may **not** use semantic resources like WordNet or word embeddings (word2vec, GloVe, etc.) for this project. The features you select must be fed to the LogisticRegression classifier in the scikit-learn Python machine learning library.
    - Things to consider:
        - whether or not to remove some or all punctuation.
        - whether to remove some or all stopwords
        - whether to normalize text to lowercase
        - whether to use techniques such as distributing negations across (some or all) subsequent text
        - whether to segment sentences
    - You must have a function that (selects and) loads features into a data structure for classification. This function should be clearly identified in comments as performing this task.
    - You must train the classifier with *random_state* set to zero (seeds the random number generator), using the ***lbfgs*** solver (works well for small datasets), and by classifying using both the *ovr* (one versus rest) and *multinomial* methods.

Note: Your features must be based only on the words in the review, as well as anything directly derivable from those words (their presence in the lexicons, position, part of speech tag, small n-grams containing the words, etc.). Do not use any external sentiment analysis tools in your solution. The cleverness of you approach counts.

- *Output* – The reviews to be classified must be found in a file named ***reviews-test.txt*** file. After each classification stage (ovr, multinomial), the program should display the following output (refer to class demo):
    - Accuracy
    - Classification Report
    - Confusion Matrix

2. (**5%**) Code comments - Add the following comments to your code:

- A section at the top of the source file(s) with the following identifying information:
    Your Name
    CSCI 6350-001
    Project #X
    Due: mm/dd/yy
- Below your name add comments in each file that gives an overview of the program or class.
- Place a one or two-line comment above each function that summarizes the workings of the function.

3. (**30%**) Analysis – Answer the following questions as a text markdown block in your code

- Selection of features – Explain how you arrived at your final set of features: what did you try? What insight lead you to them? Were there one or more that were particularly good?
- Provide your thoughts on the differences observed between one versus rest and multinomial classification.
- Provide an analysis of your measures (accuracy, f1, precision, recall) and the confusion matrix.

4. (**Up to 10%**) Extra credit – You're on your own to get these answers, and must do all parts to get credit.

- Read section 5.5 of the text book (Regularization)
- Determine how to apply regularization in sklearn.
- Discuss how you chose any regularization parameters.
- Provide a brief analysis of what regularization does, and how it affected your (unregularized) results.
- Show your code and a markdown with the analysis called for

Project submission:

You must complete this assignment using Jupyter Notebook. Submit only the .ipynb file. You may complete this on your home system, or on the NLP system. You **MUST** test it on the NLP system, where it must run without modification. Be mindful that the NLP system does not have the *handin* utility, so you'll have to submit your work from ranger.

Below is an example of how you transfer files between ranger2 and the NLP system:

On NLP System terminal - copy to ranger2: *scp test.txt ranger2.cs.mtsu.edu:~/.*

On NLP System terminal - copy from ranger2: *scp ranger2.cs.mtsu.edu:~/test.txt .*

The UP arrow in Jupyter allows the upload of a file on the local machine to where the notebook is.

Once all files are on ranger use your class C account to submit your work. The *handin* command allows assignments to be submitted electronically. The command *handin* must be followed by assignment ID and the files required. The files must be named exactly as listed here (although they may appear in any order). Submit your assignment from the directory where your source and log files are, with the following command (you may resubmit your work – it simply overwrites the previous submission. Your last submission will be graded).

handin proj4 proj4.ipynb