Lab 5: Naïve Bayes Classifier

EECS 348

Introduction

For this assignment you will implement a Naïve Bayes Classifier that analyzes the sentiment conveyed in text. When given a selection of text as input, your system will classify the text as positive or negative. The system will use a corpus of movie reviews as training and testing data, with the number of stars assigned to each review by its author as the truth data (`'1'` star is negative and `'5'` stars is positive).

Dataset

The training data is in `train.txt`, and has the following format: `ID|number of stars|review text`

You are to use the training data provided to train your Naïve Bayes Classifier. Once trained, your classifier will be evaluated by classifying unlabeled movie reviews. The test data and answers are in the files `classifyA.txt` and `answersA.txt`, respectively, with the following formats:

- classify file format: `ID|0|review text`
- answer file format: `ID|number of stars|review text`

Evaluation

Your system will be evaluated based upon the f-score of your predicted classes. Your objective is to construct the best Naïve Bayes Classifier possible for the dataset. A minimum f-score threshold will be established for a simple Naïve Bayes Classifier which uses only unigram token frequency as features. Classifiers with f-score below the minimum f-score threshold will receive zero points, while classifiers just meeting this threshold will receive 50% of the points for the test case. Two additional thresholds will be established for 70% and 100% of the points, respectively. Your system will be graded using the same training data and additional unseen test cases.

Implementation

Starter code for this assignment is comprised of `main.py` which calculates the f-score for a given test case, and makes calls to train the classifier and classify test observations, and `student_code.py` which contains the definition of a class for the Naïve Bayes Classifier. The `train` member of the classifier takes a training file as input and does not return anything. The `classify` member takes a test file with reviews to be classified and returns a python list of strings indicating the predicted class (either `'1'` for negative or `'5'` for positive) for each review in the test file. You are free to add any additional class variables, member functions or helper functions that you like. Also, you are free to manage your data in any format convenient for you. However, you are prohibited from importing any packages other than `math`, `print_function` and `student_code`. Importing other package will result in a score of zero for the assignment.

Hints

The following may be helpful when constructing your classifiers:

- the `f_score` function has code that shows one method of reading the data files,
- the probability of a review being positive given a set of features f can be calculated as:

$$P(positive \mid f) = P(positive) * \prod_{i=1}^{n} P(f_i \mid positive)$$

- since probabilities can become small, you may want to work with *log-probabilities* in order to avoid underflow (in which case products become sums),
- use add-one smoothing for tokens in the test data that were not present in the training data, and
- additional potential features to consider may include: stop-words, capitalization, review length, use of punctuation, word stems, tf-idf, etc.