
STOP!!! Please read the PDF version of the README instead of reading this file, as there are equations that aren't rendered in this file. Thank you.

HW 4 - Naïve Bayes

Introduction

In this homework, you will implement a Naïve Bayes Classifier that categorizes movie reviews as positive or negative based off of the text in the review. You will train and test your classifier on a corpus (that's ML-speak for text dataset) of movie reviews. Each movie review in the corpus has been labeled to be either postive (5-star) or negative (1-star). Only 5-star and 1-star reviews are included in the corpus.

Your Task

Implement the best possible Naïve Bayes Classifier for sentiment analysis.

The starter code in `student_code.py` contains the definition of a class for the Naïve Bayes Classifier. The `train` function of the classifier class takes a list of lines from the dataset (the format of each line is described below). The `classify` function takes another list of lines to be classified and returns a Python list of strings indicating the predicted class (1 or 5).

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` and `re`. Importing other packages will result in a score of zero for the homework.

To create the best possible classifier, you will want to consider many ways to improve the classifier. Some ways to improve the classifer include the following:

- add-one smoothing
- removing capitalization
- removing punctuation
- removing stop words
- stemming
- TF-IDF
- bigrams

There are many more ways to improve your classifier. Implement some subset of these improvements, and find some combination that creates the best classifier.

The Naïve Bayes Algorithm

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 very small, the product of these numbers can result in underflow. To get around this, use *log-probabilities* (in which case, products become sums).

Evaluation

Your objective is to construct the best Naïve Bayes Classifier possible for the dataset. To evaluate how well your classifier perfoms, we will train and test your classifier on the given dataset and calculate an F-score for each of the two classes (positive and negative). Your classifier needs to perform above a minimum F-score theshold.

Note: You may be wondering why we will both *train* and *test* your classifier; why not just *test*? That's because unlike with other ML algorithms, you will not give us some trained model parameters that we can then test by using the model to classify some test data. Instead, given how Naive Bayes "trains," we will use your code to both train and test your classifier implementation.

The provided test does a 90-10 split of the dataset, using 90% of the data for training and the other 10% for testing. We provide a function to calculate the F-score for evaluating the performance of your classifier on the test data.

Data

The dataset is in the file, `alldata.txt`, which contains about 13,000 reviews, each on its own line.

Each line of data is of the form:

```
NUMBER OF STARS|ID|TEXT
```

- The number of stars is 1 or 5.
- The text goes until a newline (`\n`).
- The text won't contain a '|', so you can safely invoke `split('|')`.

The `f_score` function has code that shows one method of reading each line of the data.

F-score

We provide a calculate of F1, an F-score that takes into account the *precision* and the *recall* of the classifier for a given class.

$f1_c = \frac{2 * p_c * r_c}{p_c + r_c}$

All tests will check the F-score for both the positive and negative classes.

Additional resources

http://facweb.cs.depaul.edu/mobasher/classes/csc575/papers/porter-algorithm.html

https://nlp.stanford.edu/IR-book/html/htmledition/dropping-common-terms-stop-words-1.html

https://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html