Project Description

Spam detector using the Naive Bayes approach

Spam Detector. You have to develop a *Python*-based spam detector using the *Naïve Bayes* approach. You can only use the following libraries: `NumPy, math, re, sys` and `Matplotlib`.

Your Python program has to be able to build a probabilistic model from the training set (available on Moodle). Your code must parse the files in the training set and build a vocabulary with all the words it contains. Then, for each word, compute their frequencies and probabilities for each class (class *ham* and class *spam*).

To process the texts, fold all characters to lowercase, then tokenize them using the regular expression `re.split('\[\^a-zA-Z\]',aString)` and use the set of resulting words as your vocabulary.

For each word w_i in the training set, save its frequency and its conditional probability for each class: $P(w_i|ham)$ and $P(w_i|spam)$. These probabilities must be smoothed using the 'add δ' method, with $\delta = 0.5$. To avoid arithmetic underflow, work in $\log_{10}$ space.

Save your model in a text file called `model.txt`. The format of this file must be the following:

1. A line counter i, followed by 2 spaces.
2. The word w_i, followed by 2 spaces.
3. The frequency of w_i in the class *ham*, followed by 2 spaces.
4. The smoothed conditional probability of w_i in the class *ham* $-P(w_i|ham)$, followed by 2 spaces.
5. The frequency of w_i in the class spam, followed by 2 spaces.
6. The smoothed conditional probability of w_i in *spam* $-P(w_i|spam)$, followed by a carriage return.

Note that the file must be sorted alphabetically. For example, your file `model.txt` could look like the following:

```
1  abc  3  0.003  40  0.4
2  airplane  3  0.003  40  0.4
3  password  40  0.4  50  0.03
4  zucchini  0.7  0.003  0  0.000001
```

Evaluating your Model. Once you have implemented your classifier, use it to train a model that can classify emails into their most likely class: *ham* or *spam*. Run your classifier on the test set given (on Moodle) and create a single file called `result.txt` with your classification results. For each test file, `result.txt` must contain:

1. a line counter, followed by 2 spaces
2. the name of the test file, followed by 2 spaces
3. the classification as given by your classifier (the label spam or ham), followed by 2 spaces
4. the score of the class *ham* as given by your classifier, followed by 2 spaces
5. the score of the class *spam* as given by your classifier, followed by 2 spaces
6. the correct classification of the file, followed by 2 spaces
7. the label right or wrong (depending on the case), followed by a carriage return.

For example, your result file could look like the following:

```
1  test-ham-00001.txt  ham  0.004  0.001  ham  right
2  test-ham-00002.txt  spam  0.002  0.03  ham  wrong
```