CSE 216 – Programming Abstractions (Spring 2019)
Programming Assignment # 2

In this assignment, you will develop a spam filter that uses word frequencies to filter spam emails. The program will be developed in Python language using PyCharm as IDE.

Machine learning is a branch of computer science consisting of algorithms and techniques for "teaching" computers to recognize patterns, make predictions, detect trends, and the like. One well-known application of machine learning is filtering spam email. As a user flags emails as spam, over time the software learns how to identify spam itself, flagging spam emails automatically.

In this assignment, you will develop a spam filter that uses word frequencies. For example, if the word "diet" appears in 63 of 500 emails flagged by the user as spam, the probability of a spam email containing "diet" is 63/500 = 0.126 or 12.6%. Such frequencies will help the software learn how to detect spam and calculate a probability that an email is spam.

File processing[1]:

Files come in two general formats: plain text files and binary files. A (plain) text file is a simple file whose contents can be read by a basic text editor. E.g. .py and .txt files are examples of text files. Everything not a text file (images, videos, MP3s, compiled programs, etc.) is called a binary file because the file has a specific structure. In this assignment, we will deal with text files. The wc function as defined below returns number of lines, words and characters in a text file:

```
def wc(filename):

    nlines = nwords = nchars = 0

    for li in open(filename):

        nlines += 1

        nwords += len(li.split())

        nchars += len(li)

    return nlines, nwords, nchars
```

We can perform a tuple assignment to save the multiple values returned by the `wc` function:

```
lines, words, chars = wc('email/good.txt')
```

[1] For details on Python file IO, see https://www.tutorialspoint.com/python/python_files_io.htm.

Dictionaries[2]

Suppose we want a distances dictionary to represent the number of feet in a single yard, fathom, furlong, or mile. We might initialize these values as follows:

distances['yard'] = 3

distances['fathom'] = 6

distances is now: {'fathom': 6, 'yard': 3}

The strings, ‘fathom’, and ‘yard’ are the keys of the dictionary.

6 and 3 are the values of the dictionary.

Word frequencies for Spam filtering

Getting back to our original problem, we want to build a program that will do basic spam filtering. Part of the solution will include counting how many times each word appears in the input email message. We can define a dictionary called count to serve this purpose:

```
count = {}
```

To increment the count for a word, we can use the += 1 notation. Suppose the variable word has the string we want to increment the count of. We can write this as follows:

```
count[word] += 1
```

Stripping punctuation is a common operation in text processing, Python has it built-in through the string module:

import string

```
s1 = 'Good morning!'

s2 = s1.strip(string.punctuation)

s2 will contain 'Good morning'
```

Spamicity (aka Spaminess)

Now that we have a way of counting the number of occurrences of each word in a file, we can use it to help us calculate the probability that an email is spam. Suppose we know that the word “secret” appears in 252 out of 1000 spam messages. We might define the spam probability of “secret” as 252/1000 = 0.252. In other words, the probability of seeing the word “secret” in a piece of spam is 0.252.

This idea of a probability of some event being based on some known fact is called conditional probability. The probability of seeing a word w in an email we know is spam will be denoted P(w/spam). Read this as “the probability of seeing word w, given a spam email”.

[2] For details on Python dictionaries, see https://www.tutorialspoint.com/python/python_dictionary.htm.

Refer to the two txt files provided along with this assignment: email/good.txt and email/bad.txt. Each row of these files contains two values: probability and word. The file good.txt shows the probabilities of words appearing in emails which are not spam. The file bad.txt shows the probabilities of words appearing in emails which are spam.

Spamicity

You will define a function load_probabilities which will take as input text files bad.txt and good.txt and will return a dictionary specifying probability of a word as follows:

```
pbad = load_probabilities('bad.txt')

pgood = load_probabilities('good.txt’)
```

pbad is a dictionary that tells us the probability of a word appearing in a spam message. Likewise, pgood is a dictionary that tells us the probability of a word appearing in a non-spam message. For example, pbad['money'] is 0.127 and pgood['money'] is 0.0164. We see that the probability of “money” appearing in a spam message is 0.127, and its probability of appearing in a non-spam message is 0.0164. If we encounter a word in an email that is not in either dictionary, then, we really don’t know anything about the word and can’t use it to help us identify spam messages.

With pbad and pgood we can now define the “spamicity” of a word. The spamicity will be closer to 1 than to 0 when an word appears in more spam messages than good messages. The spamicity will be closer to 0 than to 1 when a word is found in more good messages than in spam messages.

Define spamicity of a word w using this formula:

```
spamicity(w) = P(spam|w)

                 = P(w|spam) / (P(w|spam) + P(w|good))
```

This formula is based on a concept called Bayesian inference[3].

The two conditional probabilities in the formula will come directly from the pbad and pgood dictionaries. You will define a function spamicity to compute spamicity as follows:

```
def spamicity(w, pbad, pgood):

    if w in pbad and w in pgood:

        return pbad[w] / (pbad[w] + pgood[w])

    else:

        return None
```

For example, spamicity(“money”) is 0.89, meaning that we predict 89% of incoming messages containing the word “money” are spam. If the word w is not in one or both dictionaries, the value None is returned.

[3] For the details of Bayesian inference, see https://towardsdatascience.com/probability-concepts-explained-bayesian-inference-for-parameter-estimation-90e8930e5348.

Identifying junk mail

You will use spamicity function to help classify entire emails as good or spam. Somehow, we need to combine the spamicity values of the words in an email message. The approach we will take is to consider "interesting" words – those words with high or low spamicity.

Let's define the "interestingness quotient" (IQ) of a word w as IQ(w)=|0.5-s|, where s is the spamicity of word w. The IQ of a word will range from 0.0 to 0.5, with 0.5 meaning a very interesting word.

Consider some examples:

If s=0.9, then |0.5-0.9|=0.4

If s=0.05, then |0.5-0.05|=0.45

A "boring" word would have s near 0.5. Consider s=0.47. Then, that word's IQ is |0.5-0.47|=0.03, which is quite low.

You will build a data structure called a priority queue using PriorityQueue class, which allows add and remove items from a collection, always putting the highest priority item at the front. There is a limit on the number of elements to be stored in PriorityQueue. The priority queue should keep track of the most interesting words in an email where the words are sorted by their IQs. As we add or remove words, the most interesting word will always be at the front. The insert function of PriorityQueue takes as input a word and its spamicity. On inserting a word, the PriorityQueue class should compute the IQ of the word and then update the PriorityQueue. Here's a short example of how we might use the queue:

```
pq = PriorityQueue (10) # creates a queue to hold 10 words

s = spamicity('there', pbad, pgood)

pq.insert('there', s)

s = spamicity('book', pbad, pgood)

pq.insert('book', s)
```

The PriorityQueue class will also have a function getSpamicityList() which will return a list of spamicity values of the words in a queue. E.g.

```
spamicitylist = pq.getSpamicityList()
```

Now you will define the top-level function pspam, which will give us a probability that a particular message is spam. The input will come from a text file containing email message. The function will depend on another helper function called combined_probability that uses some formulas from probability theory to combine all the word spamicity values into a single number.

Given a list of word spamicity values, the combined_probability function is defined as follows:

```
def combined_probability(spamicitylist):

    p = q = 1.0

    for x in spamicitylist:
        p *= x
        q *= (1.0 - x)
    return p / (p + q)
```

Following is the pseudocode for pspam function:

```
fun pspam(emailmsg):
     Initialize priority queue of size 15.
     Load probabilities from 'bad.txt' file in a dictionary pbad.
     Load probabilities from 'good.txt' file in a dictionary pgood.
     Open emailmsg file and for each word do the following:
     Calculate spamicity of the word.
          If spamicity of a word is not None then insert word and
corresponding spamicity in a priority queue.
    return combined probability of words in a queue.
```

Your main program named spamdetector.py will accept a command line argument[4] which is a path to the file containing email message. The program will invoke pspam function with filename and classify email into one of the following categories:

Combined probability > 0.7 -> high probability of spam

Combined probability > 0.5 -> probably a spam

Combined probability > 0.3 -> probably not a spam

Combined probability > 0.1 -> high probability of not a spam

The program will also print contents of message.

1) **Submission:** Submit the following .py files (Do not submit zip file, but submit Python files individually):
    a. spamdetector.py that contains:
        i. load_probabilities function
        ii. spamicity function
        iii. combined_probability function
        iv. pspam function
        v. Should accept message filepath as command-line argument
        vi. Should import PriorityQueue class from queueimpl.py file
    b. queueimpl.py that contains:
        i. Definition of PriorityQueue class
        ii. insert function

[4] See the following for Python command line arguments:
https://www.tutorialspoint.com/python/python_command_line_arguments.htm

iii. getIQList function

2) **Rubric:**
    a. Valid submission containing above files and functions (5 points)
    b. Properly working load_probabilities function (5 points)
    c. Properly working pspam function (7 points)
    d. Should accept message filepath as command-line argument (3 points)
    e. Should import PriorityQueue class from queueimpl.py file (3 points)
    f. Proper definition of PriorityQueue class (7 points)
    g. Properly working insert function (5 points)
    h. Properly working getIQList function (5 points)
    i. Proper output (10 points)

3) **Evaluation:** A slot will be announced for evaluating the assignments. During this time the instructor/TA will download and run your program and it will be checked for correctness using a few use-cases. If required, instructor/TA will contact you for clarifications.

Example 1:

➢ python spamdetector.py 'email/msg1.txt'

Result: 0.92930483265 (high probability of spam)

File contents:

Hurting for funds right now?

It doesn't have to be that way. Here is 1,500 to ease your pain:

http://bulk.hideorganic.com/171026390236329103248372180

Transfer immediately to the account of your choice:

http://bulk.hideorganic.com/171026390236438808248372180

Take your time to pay off this amazing loan. Small payment due in late September

or early October (and not all in one payment!).

Example 2:

➢ python spamdetector.py 'email/msg2.txt’

Result: 4.400695206e-05 (high probability of not a spam)

File contents:

Hi John:

Interesting that the key might be preventing ANY crystals from being able to nucleate - which kicks off a chain reaction and the whole thing goes to hell. Thus the very clean pot and not allowing anything to splash up onto the sides. Cooking really is chemistry!

Thanks for the links.

Susie

[Rest of the message]

Example 3:

- ➢ python spamdetector.py 'email/msg3.txt'

Result: 0.05810198935 (high probability of not a spam)

File contents:

Guess what conery@cs.uoregon.edu!

AUTO CLEARANCE ENDS TONIGHT! : Price Drop On All Vehicles

Want To Drive A Brand New Car Today For A Fraction Of What You Thought You Would Pay?

Now You Can!

Dealers Have Drastically Reduced MSRPs.

AVAILABLE ONLY UNTIL 10:00 PM TONIGHT!

http://server.beavercreekdistrict.com/3813010413df842258012632451675

Click this link to unsubscribe: http://server.beavercreekdistrict.com/3813010413df528010632451675

Example 4:

- ➢ python spamdetector.py 'email/msg4.txt'

Result: 3.758445e-15 (high probability of not a spam)

Hi John,

I meant to ask you if you tried the revised cat command. Were you able to do what you needed?

Regarding your lab meetings... sure, I could come and give a brief description and answer any questions your group members might have. My assistant, Erik, has just put up more information from Chris' slides onto the wiki that might be helpful. It would be helpful to me if I knew in advance more specifically what kind of questions to address before coming - perhaps you can collect some at today's group meeting?

Cheers,

Rob

Note:

If your code does not compile, it will not be graded.

Late submissions will not be accepted under any circumstances.

To be safe, always, ALWAYS, prepare to submit ahead of time, not exactly AT last moment!

Submission deadline: Monday 15 April, 11:59 PM