CSE 231
Spring 2009

Programming Project 7

This assignment is worth 50 points (5% of the course grade) and must be completed and turned in before midnight on Monday, March 16th . That seems like two weeks but the second one is spring break so it might be to your advantage to get this project done early. It is a bit of work.

This assignment is an exercise in the use of dictionaries and functions.

Background
In this project, we will do some more work on text processing. You will explore the n-gram concept in Natural Language Processing. N-grams are sequential patterns of n-words that appear in a document. To simplify the project, we are just considering unigrams and bigrams. Unigrams are the unique words that appear in a text whereas bigrams (bi-grams) are patterns of two-word sequences that appear together in a document.

Essentially, unigrams are word counts, the number of times a word occurred in documents in the data set being examined. Bigrams are the number of times a two-word sequence has occurred in documents of our data set. Consider the following short document:

```
This is a test
This is only a test
A test should not cause concern
Is this test difficult?
```

First, we note that case does not matter. Thus "Test" and "test" are equivalent for our purposes. For unigrams (word counts), we have the following:

```
"test":4, "this":3, "is":3, "a":3, "only":1, "should":1,
"not":1, "cause":1, "concern":1, "difficult":1
```

For bigrams, we look for two word sequences in either order. That is, the two words "operating systems" or "systems operating" should count as the same bigram. With that in mind, we get the following bigrams from our text:

```
"this is":3, "is a":1, "a test":3, "is only":1, "test should":1,
"should not":1, "not cause":1, "cause concern":1, "this test":1,
"test difficult":1
```

Note the count of the bigram "this is" is 3, since "This is" occurred on lines 1 and 2 and "Is this" occurred on line 4. Also note that we typically process documents to remove common words such as "a" or "the" as they contribute little to the uniqueness of a document.

In the end, we can use the statistics of bigram occurrence vs. unigram occurrence to tell something about the uniqueness of a document.

Assignment Overview
You will are given a dataset called "web.txt" which is a collection of web pages that were extracted from the webkb publically available dataset (http://www.cs.cmu.edu/~WebKB/) . The goal of webkb is to provide a common set of documents to test the ability of search engines.

We did some preprocessing on this dataset and selected a subset of webkb for this project. You are required to find all the unigrams and the bigrams in the data set and display the contingency table (described below) for a selected number of bigrams which are given in "bigram.txt"; a file that we provided for you.

Task
There are three main components of this project. Each component is to be written as a separate function (that means three functions, at a minimum).

1. You are given a text file "web.txt" which contains a list of documents. Each row in the file is basically a web page processed in various ways to identify the unique words in the document. Note that it is all lower-case already! The file is not very readable, but that doesn't matter for this process. You are required to read each document (that is, each line) in the file and extract all the unigrams and bigrams in all of the documents (all of the lines) in the file. Save the unigrams and bigrams in two separate dictionaries.
2. Read in the contents of the file "bigrams.txt". Each line of the file consists of a pair of words separated by a space. Store these pairs in a data structure of your choice.
3. Produce the contingency table for each bigram pair you that you read from the "bigrams.txt" file. The contingency table of a bigram is a 2x2 table that displays the frequency information of a given bigram and the frequency of the individual words (the unigrams). Here is a typical contingency table. It is for the pair "programming" and "systems". The ~ symbol means that the word following **does not occur**:

	systems	~systems
programming	3	54
~programming	21	11650

The table above shows a contingency table for the bigram ("programming", "systems"). The statistics in the table are computed as follows.

1. the count of bigrams in the file having the words "programming" and "systems" (in either order) is 3.
2. the count in the file of the unigram "programming" minus the number of times "programming" and "systems" occurred as a bigram. Here, there were 57 occurrences of "programming", minus the three for the bigram, making 54.
3. the count in the file of the unigram "systems" minus the number of times "programming" and "systems" occurred as a bigram. Here, there were 24 occurrences of "programming", minus the three for the bigram, making 21
4. the total number of bigrams in the file not having either the word "systems" or "programming" in it. The sum of all the occurrences of all of the bigrams in the entire dataset will equal the sum of 1,2, 3 and 4 above. In this example the sum of all bigram

occurrences is 11728.

Program Specifications:

1) In the first function, read the text file 'web.txt' that is given. You need to process each line in the file individually in order to keep track of the words that appear in each document. Discard the document number at the beginning of each row. Use two different dictionaries: one to save the unigrams and the other to store the bigrams. A tuple should be used as the key for the bigrams dictionary.
2) Read "bigrams.txt" to get the set of bigrams for which you are going to report the contingency table.
3) Write a function that creates and prints the contingency table.

Hints:

When computing the bigrams, do not consider order. For example, the bigram (program,system) is the same as the bigram (system,program), so only one of these entries should appear in the dictionary, and either should update the count for this entry.

Deliverables

You must use Handin to turn in the file proj07.py – this is your source code solution; be sure to include your section, the date, the project number and comments describing your code. Please be sure to use the specified file name, and save a copy of your proj07.py file to your H: drive as a backup.

Sample output of the program:

```
Bigram counts:  94785
                     operating  ~operating
   practicum               1            4
  ~practicum              49        94731

------------------------------------------------------------

                     cornell      ~cornell
       upson              23           35
      ~upson             937        93790

------------------------------------------------------------

                     exam            ~exam
    midterms              1            5
   ~midterms            124        94655

------------------------------------------------------------

                     home            ~home
         src              2           52
        ~src            301        94430
```

```
-----------------------------------------------------------------

                     systems      ~systems
 programming               6           364
~programming             121         94294

-----------------------------------------------------------------

                     home           ~home
   frontpage              1             0
  ~frontpage            302         94482

-----------------------------------------------------------------

                     science      ~science
        fall               8           133
       ~fall             142         94502

-----------------------------------------------------------------
```