NLP Lab Session Week 4
September 18, 2013

Mutual Information, Stemming and Lemmatization, Reading files

Getting Started

In this lab session, we will use two saved files of python commands and definitions and also work together through a series of small examples using the IDLE window and that will be described in this lab document. All of these examples can be found in python files on the blackboard system, under Resources.

Bigram.py, processtextfile.py, Labweek4examples.py, desert.txt, Smart.English.stop

Put all of these files in one directory (and remember where that is!) and open an IDLE window.

Module of Bigram Distribution and Mutual Information functions

To make it easier to use the frequency distribution functions that we have defined, they are all placed into one python file, which python would call a Module. The name of the file is Bigram.py and if the file is in the same directory as another Python program, you can say "import Bigram" to access the function definitions. Then you use the name of the module to use the function names, for example, Bigram.bigramDist.

In your IDLE window, use the File menu to open the Bigram.py file. Now if you want to use the functions in the Bigram file in your IDLE window, you will need to add the path to the Bigram file to the system path.

To see which directories the system will search to find programs for the IDLE window:
>>> import sys
>>> sys.path

Suppose that you added the Bigram.py file to a directory on the H: drive called NLPclass. To add that directory to your path:
>>> sys.path.append('H:\\NLPclass')

Now you could use the functions such as bigramDist in your IDLE window without copying the definition to the window.

Mutual Information

Review the definition of association ratio from the Church and Hanks paper, and the mutualinfo function. I have put this function, along with the other unigram and bigram frequency distribution functions into the Bigram.py file.

From your IDLE window under the File menu, use Open to open the Bigram.py file into another IDLE window. Read through this module through to see the function definitions.

Now we will again get the text from the novel Emma as an example to work on. These examples are in the file labweek4examples.py.

```
>>>import nltk
>>>nltk.corpus.gutenberg.fileids( )

>>>file0 = nltk.corpus.gutenberg.fileids( ) [0]
>>>ematext = nltk.corpus.gutenberg.raw(file0)
>>>ematokens = nltk.wordpunct_tokenize(ematext)
>>>emawords = [w.lower( ) for w in ematokens]
```

In order to call the mutual information function, we can

```
>>> import Bigram
```

Let's call the function with an empty stop word list and a threshold of 2:

```
>>> MIdist = Bigram.mutualinfo ( emmawords, [] , 2)
```

Try looking at the keys and mutual information scores for the top 20 scoring bigrams.

```
>>> for pair in MIdist.keys() [:30]:
    print pair, MIdist[pair]
```

Note the differences between the mutual information lists and the bigram lists that we got previously.

[Note that if you make changes to the Bigram module, you should save it and then use the reload function, instead of importing it again.

```
>>> reload(Bigram)
```

The above examples used a threshold of 2, but Church and Hanks actually recommend a threshold of 5.

Stemming and Lemmatization

For this part, we will use the emma text from the Gutenberg Corpus as we did above.

Note that emmatokens has words with regular capitalization and emmawords has lower-case words with no capitalization.

NLTK has two stemmers, Porter and Lancaster, described in section 3.6 of the NLTK book. To use these stemmers, you first create them.

```
>>>porter = nltk.PorterStemmer()
l>>>ancaster = nltk.LancasterStemmer()
```

Then we'll compare how the stemmers work using both the regular-cased text and the lower-cased text.

```
>>>emmaregstem = [porter.stem(t) for t in emmatokens]
>>>emmaregstem[1:100]

>>>emmalowerstem = [porter.stem(t) for t in emmawords]
>>>emmalowerstem[1:100]
```

Try the same examples with the Lancaster stemmer.

The NLTK suggests that we try building our own simple stemmer by making a list of suffixes to take off.

```
def simplestem(word):
    for suffix in ['ing', 'ly', 'ed', 'ious', 'ies', 'ive', 'es', 's', 'ment']:
        if word.endswith(suffix):
            return word[:-len(suffix)]
    return word

#try the above stemmer with 'friends'
stemmedword = simplestem('friends')
stemmedword
```

The NLTK has a lemmatizer that uses the WordNet on-line thesaurus as a dictionary to look up roots and find the word.

```
wnl = nltk.WordNetLemmatizer()
emmalemma=[wnl.lemmatize(t) for t in emmawords]
emmalemma[1:100]
```

Processing Text from Files

(See if we can open another IDLE window to do this part.)
Open the processtextfile.py and read through it to observe how it reads text from the file desert.txt and how it creates a stop word list from the file Smart.English.stop. Look at the Smart.English.stop words file in a text editor. Note that processtextfile then calls the mutualinfo function, which computes a bigram distribution with the association ratio measure computed in a window of 2.

Run the processtextfile by selecting Run Module from the Run menu. The results will appear in your main IDLE window (which restarts the session and erases all the definitions that you already made).

Modify the code to try the mutual information function with and without stopwords, by using the comment character # to switch between the two statements. Also try thresholds of 3, and perhaps 5.

Note that another list of stopwords is available in NLTK: nltk.corpus stopwords. This list has 127 words and can be accessed as follows:

```
>>> from nltk.corpus import stopwords
>>> nltkstoplist = nltk.corpus.stopwords.words('english')
>>> nltkstoplist
```

For your homework, if you develop a file similar to the processtextfile in lab, then you can rerun your examples without as much typing into IDLE.

Note that you can also run this python file by using python directly from a terminal/command prompt window. That is, on a PC, open a command prompt window and change directory until you get to the LabExamples4 directory. Then run (I think):

```
%  processtextfile.py
```

On a Mac, open a terminal window and change directory until you get to the LabExamples4 directory. Then run:

```
% python processfile.py
```

Ideas for Homework

For documents that come from NLTK corpora, read the NLTK book sections from Chapter 2 on the different corpora.

Also note that Chapter 2 discusses how to load your own text with the PlainCorpusReader, or you can just read text from files as we did in the lab example with desert.txt.

Example of using word frequencies to analyze text:
Nate Silver's analysis of State of the Union Speeches from 1962 to 2010.
http://www.fivethirtyeight.com/2010/01/obamas-sotu-clintonian-in-good-way.html
Here is a statement of the question that he is trying to answer by looking at word frequencies to compare the SOTU speech in 2010 with earlier speeches:
"What did President Obama focus his attention upon and how does this compare to his predecessors?"

[And you may find it interesting to note also the criticism of the technique at http://thelousylinguist.blogspot.com/2010/01/bad-linguistics-sigh.html. I am not expecting you to be experts in how you categorize the words or bigrams that you use in your analysis (and I actually think that Nate Silver was not too far off the mark in picking categories of words to answer his question.)]

How is this example different from your assignment? It is different because it does more comparisons that you are required to do and only uses word frequencies while you must also look at bigram frequencies and mutual information. But if you choose the analysis option, this is the type of thing that you are aiming for. Also, you can just use word lists and frequencies/scores without making colored graphs.

Also note that you can collect your own text from just about anywhere. For a (too short) example of this, see the potato chip marketing example.

Discuss the programming option.

There is no exercise for today.