NLP Lab Session Week 3
September 15, 2011

**Mutual Information, Stemming and Lemmatization, Reading files**

**Installing NLTK Data**

Reinstall nltk-data if needed.

**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, Labweek3examples.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 bigram 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**

Look at the paper by Church and Hanks and observer their definition of the Association Ratio. I have written a function for that, with a window of 2, called mutualinfo, and put it 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 function to see how it implements the definition.

Now we will again get the text from the novel Emma as an example to work on. These examples are also in the file labweek3examples.py.

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

>>>file0 = nltk.corpus.gutenberg.fileids( ) [0]
>>>emmatext = nltk.corpus.gutenberg.raw(file0)
>>>emmatokens = nltk.wordpunct_tokenize(emmatext)
>>>emmawords = [w.lower( ) for w in emmatokens]
```

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 ( words, [] , 2)
```

Try looking at the keys and mutual information scores for the top 20 scoring bigrams.
```
>>>for pair in MIdist.keys() [:20]:
    print pair, MIdist[pair]
```

**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 stem(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=stem('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**

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. Note that it 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.