

NLP Lab Session Week 3  
February 4, 2010

## **Bigram distributions and Association Ratio**

### **Installing NLTK Toolkit**

Reinstall nltk-2.0b7.win32.msi and Copy and Paste nltk\_data from H:\nltk\_data to C:\nltk\_data, or download nltk\_data.zip from the server, given the URL in class. Note that this is just the book data, which should be all we need for labs.

### **Getting Started/Processing Text**

In this session, we will complete the bigram distribution and association ratio functions that we didn't get to last time.

Start your Python IDLE window.

Type in

```
>>> import nltk
```

In this lab session, we will work together through a series of small examples using the IDLE window and that will be described in this lab document. However, for purposes of using cut-and-paste to put examples into IDLE, the examples can also be found in a set of python files on the iLMS system, under Resources.

LabWeek3examples.py, SimpleBigram.py,

Open an IDLE window. Use the File-> Open to open the labweek5examples.py file. This should start another IDLE window with the program in it. Each example line can be cut-and-paste to the IDLE window to try it out.

First, we want to set up text and process it. Go to the file with LabWeek3examples.py and copy and paste the following lines to get started. This gets the text of the book Emma, separates it into tokens with the wordpunct tokenizer, and converts all the characters to lower case.

```
>>>print 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]
```

For purposes of the lab, we create a list with only the first 101 words:

```
>>>shortwords = emmawords[11:111]
```

Recall that we can make a frequency distribution of words.

```
>>> from nltk import FreqDist
```

```
>>>shortdist = FreqDist(shortwords)
>>> shortdist.keys( )
```

Or we can look at the frequencies of all the words in the distribution:

```
>>> for word in shortdist.keys():
    print word, shortdist[word]
```

## Review Bigram distribution

Next we are going to create a frequency distribution to count all the bigrams of a set of words.

One way to create the bigram frequencies is to start with an empty Frequency Distribution, iterate over the words pairs (bigrams) and increment the count of each word pair in the distribution.

Since you may want to use the definitions of these functions, we have separated them into different python files. For this function, use 1SimpleBigram.py, and copy and paste the definition of the bigramDist function:

```
>>> def bigramDist(words):
    biDist = FreqDist()
    for i in range(1, len(words)):
        biword = words[i-1] + ' ' + words[i]
        biDist.inc(biword)
    return biDist
```

And then we can apply the function to the short emma words:

```
>>> shortbidist = bigramDist(shortwords)
```

Here's one way to view the distribution with keys and values:

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

Or just the ones with frequency greater than 1:

```
>>> for pair in shortbidist.keys():
    if shortbidist[pair] > 1:
        print pair, shortbidist[pair]
```

## Review Regular Expressions

Now we'll tackle the problem with getting rid of the words which are really just punctuation or special symbols. For this, we use a regular expression pattern that will match word that contains any character which is not lowercase alphabetical. We first must import the regular expression package from Python (re).

```
>>> import re
```

Define a pattern p that will match any words that contains non-alphabetical characters.

```
>>> p = re.compile('[^a-z].*')
```

The function p.match tests if a word matches, e.g. the symbol "-" matches because it is non-alphabetical.

```
>>> m = p.match('-')
```

```
>>> if m: 'matched non-alphabetical'
```

## Bigram Frequency Distribution with only Alphabetical (A) words

We'll add this to our bigram distribution to select only alphabetic words.

```
>>> def bigramDistA(words):
    biDistA = FreqDist()
    p = re.compile('.*[a-z].*')
    for i in range(1, len(words)):
        m0 = p.match(words[i-1])
        m1 = p.match(words[i])
        if m0 or m1: continue
        biwordA = words[i-1] + ' ' + words[i]
        biDistA.inc(biwordA)
    return biDistA
```

```
>>> shortbidistA = bigramDistA(shortwords)
>>> shortbidistA.keys()
```

Now let's test this function on all the Emma text:

```
>>> bigDist = bigramDistA(emmawords)
>>> bigDist.keys()[:99]
>>> bigDist['to be']
```

### **Bigram Frequency Distribution with Alphabetical (A) words and filtered by Stopwords (S)**

This shows another problem, which is that our most common words are all the simple words. One way to work around this is to make a list of common words to ignore: the stop word list. Here is a simple start to a stop word list.

```
>>> stopwords = ['to', 'be', 'of', 'the', 'in', 'it', 'was', 'i', 'am', 'she', 'had', 'been', 'is', 'have', 'could',
'not', 'her', 'he', 'do', 'and', 'would', 'such', 'a', 'his', 'must']
```

When we want to add stop words to the list, we can use append for one word, or extend for a list of words.

```
>>> stopwords.append('all')
>>> stopwords
```

In our next version of the bigram distributions, we will not count any bigram that contains a word on the stop word list

```
>>> def bigramDistAS(words, stoplist):
    biDistAS = FreqDist()
    p = re.compile('.*[a-z].*')
    for i in range(1, len(words)):
        m0 = p.match(words[i-1])
        m1 = p.match(words[i])
        m2 = words[i-1] in stoplist
        m3 = words[i] in stoplist
        if m0 or m1 or m2 or m3: continue
        biwordAS = words[i-1] + ' ' + words[i]
        biDistAS.inc(biwordAS)
    return biDistAS
>>> shortbiDistAS = bigramDistAS(shortwords, stopwords)
>>> shortbiDistAS.keys()
```

### **Bigram Frequency Distribution, Alphabetical (A) words, Stopwords (S) and which are above a frequency Threshold (T)**

As a final version of our bigram distribution functions, we can add the idea of having a frequency threshold; that is, we will not count any bigram that contains a word that is infrequent in the corpus, where infrequent means that its frequency is less than the threshold.

```
def bigramDistAST(words, stoplist, threshold):
    biDistAST = FreqDist()
    uniDist = FreqDist(words)
    p = re.compile('.*^[a-z].*')
    for i in range(1, len(words)):
        m0 = p.match(words[i-1])
        m1 = p.match(words[i])
        m2 = words[i-1] in stoplist
        m3 = words[i] in stoplist
        m4 = uniDist[words[i-1]] < threshold
        m5 = uniDist[words[i]] < threshold
        if m0 or m1 or m2 or m3 or m4 or m5: continue
        biwordAST = words[i-1] + ' ' + words[i]
        biDistAST.inc(biwordAST)
    return biDistAST
```

```
>>> shortbiDistAST = bigramDistAST(shortwords, stopwords,2)
>>> shortbiDistAST.keys()
['with very']
```

Now we can test this on our entire text of Emma.

```
>>> emmaDistAST = bigramDistAST(emmawords, stopwords, 5)
>>> emmaDistAST.keys()[:50]
```

Or we can view the key, frequency pairs for the top frequency keys

```
>>> for k in emmaDistAST.keys()[:50]:
    print k, emmaDistAST[k]
```

This bigram frequency function is what we need to use in the Association Ratio, sometimes also called Mutual Information, that we will use in Homework 1.

Continue the Exercise from Week 2:

For this exercise, you may work in groups of 2-3, or you can choose to do your own work.

- Choose a file that you want to work on, either one of the files from the book corpus, or one from the Gutenberg corpus. You may use the one that you used last time, or choose a different one.
- Run the different bigram frequency distribution functions on your corpus and look at the top 20 keys from each of them: BigramDist, BigramDistA, BigramDistAS, BigramDistAST. Do you see any improvements that should be made to these distributions?

To complete the exercise, choose one of your top 20 frequency lists to report to show to the class. Write an introductory sentence of paragraph telling what text you chose and what bigram distribution function you chose. Put this and the frequency list in a discussion posting in the iLMS system under IST 664.