NLP Lab Session Week 2
September 3, 2014

Starting a Python and an NLTK Session

Open a Python 2.7 IDLE (Python GUI) window by going to All Programs->Python 2.7 -> IDLE (Python GUI).

You will probably want to work by having the IDLE window open for testing NLTK and a browser window open with these instructions. You may also want to have a separate tab or window open to the NLTK book: http://www.nltk.org/book_1ed/, where these examples are taken from Chapter 1.

In the following, examples for you to try are given following the Python Idle prompt of >>>. You can copy and paste the Python example into the Idle window, or you can type the example in.

Python and NLTK Resources

Python WikiBooks: http://en.wikibooks.org/wiki/Python_Programming
Python tutorial from python.org: http://docs.python.org/2/tutorial/

NLTK book: http://nltk.org/book_1ed/

We will take a quick look at lists in Section 1.2 that was your assigned coursework for this past week.

Counting Frequencies of Words

To get started, we'll repeat some of the steps that we did in the last lab to obtain some text examples from the Gutenberg Corpus available in the NLTK.

```
>>> import nltk
```

For purposes of this lab, again we will work with the first book, Jane Austen's "Emma". First, we get the fileid for Emma and then get the raw text as a string.

```
>>> file1 = nltk.corpus.gutenberg.fileids( ) [0]
>>> emmatext = nltk.corpus.gutenberg.raw(file1)
>>> len(emmatext)
```

Since this is quite long, we can view part of it, e.g. the first 120 characters
```
>>> emmatext[:120]
```

NLTK has several tokenizers available to break the raw text into tokens; again we will use one that separates by whitespace, punctuation and other special characters:

```
>>> emmatokens = nltk.word_tokenize(emmatext)
>>> len(emmatokens)

>>> emmatokens[:50]
```

Let's decide that we want to count upper case words such as "They" to be the same as a lower case "they", and to simplify things, we won't worry about proper names. So we convert all the tokens to their lower case version and call them words:

```
>>> emmawords = [w.lower( ) for w in emmatokens]
>>> emmawords[:50]
>>> len(emmawords)
```

[Side note on Python: Here we defined emmawords by using a Python list comprehension. It would be equivalent to define emmawords by starting with an empty list and using a "for" loop that kept appending the lower case of each word in emmatokens .

Note that a Python "for" loop can be used to iterate over every element of a list.

Note the special syntax of Python for a multi-line statement: The first line must be followed by an extra ":", and each succeeding line must be indented by some number of spaces (as long as they are all indented by the **same** number of spaces.) Just hit enter to finish the statement.

```
>>> emmawords = []
>>> for w in emmatokens:
   emmawords.append(w.lower())

>>> emmawords[:50]
```

End note.]

Python Frequency Dictionary

We will start to create a frequency list using a Python dictionary, as a learning exercise about Python dictionaries. In the next section, we'll use the NLTK Frequency Dictionary module instead.

Python dictionaries are described in the NLTK book, in section 5.3. Using dictionaries to count frequencies of words is described in section 1.3.

We will create a dictionary that has (key, value) pairs where the key is the word and the value is the frequency. We start with an empty dictionary.

```
>>> emmadict = {}
```

We will then write a for loop to go through the words and count them. To do that, if the word is already in the dictionary, we add 1 to its count and if not, we start the count at 1.

Note that we indent for the “for” loop and then we indent some more for the body of the “if” and the “else” parts.

```
>>> for w in emmawords:
   if w in emmadict:
      emmadict[w] += 1
   else:
      emmadict[w] = 1
```

How many unique words were counted? We can get the number of keys in the dictionary, where the keys are given by calling the keys() function on the dictionary.

```
>>> len(emmadict.keys())
7344
```

We can look up individual words and print the frequency:

```
>>> emmadict['the']
5201
>>> emmadict['of']
4291
>>> emmadict['a']
3129
```

We don’t really want to print the frequencies of all 7,000 words, but we can print 30 of them, using the first 30 keys, but we note that the keys of a dictionary don’t occur in any particular order:

```
>>> for word in emmadict.keys()[:30]:
   print word, emmadict[word]
```

Now what we really want is the list of word frequencies in ranked order, i.e. the most frequent first. We can do that in Python, but it’s a little awkward, so NLTK has conveniently defined a class to do that for us.

NLTK Frequency Distributions

NLTK has a set of functions that use a data structure called a Frequency Distribution, FreqDist.

This structure is an extension of the Python dictionary structures. We can import it from the nltk probability module.

```
>>> from nltk.probability import FreqDist
```

This class allows you to make a Frequency Distribution just by initializing it with a list of words. It will do all the counting for you and create a distribution in which the set of keys

are all the words, and the set of values are the frequency (count) of each word. The keys( ) function produces the list of words in order of decreasing frequency.

```
>>> fdist = FreqDist(emmawords)
>>> fdist.keys( )[:50]
```

We can treat the frequency distribution just like a Python dictionary and we can look at the frequencies of individual words:

```
>>> fdist['emma']
>>> fdist['the']
```

Or we can use a for loop to look at the frequencies of the first words, and this time the keys are sorted in the order of decreasing values.

```
>>> for word in fdist.keys()[:30]:
   print word, fdist[word]
```

Making a Python Program

Suppose that we want to look at the frequency distribution of one of the other Gutenberg books. It would be nice if we could save the steps of getting the raw text from the book file, getting the words and putting them into a Frequency Distribution. First look at the books available from Gutenberg.

```
>>> nltk.corpus.gutenberg.fileids( )
['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-kjv.txt', 'blake-
poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt', 'carroll-alice.txt', 'chesterton-
ball.txt', 'chesterton-brown.txt', 'chesterton-thursday.txt', 'edgeworth-parents.txt',
'melville-moby_dick.txt', 'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-
hamlet.txt', 'shakespeare-macbeth.txt', 'whitman-leaves.txt']
```

In the IDLE window, we'll create a new module (or new Python shell) and type in the steps, with a variable that we can change to get different files/books.

```
# program to get text from Gutenberg files
import nltk
from nltk.probability import FreqDist

fileno = 1

nfile = nltk.corpus.gutenberg.fileids( ) [fileno]
ntext = nltk.corpus.gutenberg.raw(nfile)
print 'First 200 characters of text', ntext[200]
print

ntokens = nltk.word_tokenize(ntext)
print 'First 100 tokens', ntokens[:100]
print
```

```
nwords = [w.lower( ) for w in ntokens]

ndist = FreqDist(nwords)

print '30 Top Frequency Words:'
for word in ndist.keys()[:30]:

   print word, ndist[word]
```

After putting this into the new module, if we select Run Module, it will restart the python shell and run these python steps. Warning: when it restarts the python shell, it loses all the definitions that you did previously.

Try it out:

The text that we first imported from NLTK book are already separated into lists of words. Create a frequency distribution for the words in text1 from the NLTK book (Moby Dick) by applying FreqDist directly to text1. For example:

```
>>> from nltk.book import *
>>> mbdist = FreqDist(text1)
```

Option 1: Use text1 as shown or pick one of the other texts and make a FreqDist. Print out some portion of the keys and pick several words and look at the frequencies.

Option 2: Pick one of the Gutenberg texts, run the module with that number and look at the results. Add some statements to print frequencies of some other individual words.

Exercise to submit this week:

Go to Blackboard and find the Discussion for the second week exercises. Create a post in which you:

State which text you used in the “try it out”, give some of the top keywords and give the frequencies of two of the words not on the top frequency list.