NLP Lab Session Week 4
September 23, 2015

Reading Text from Files, Stemming and Lemmatization

Getting Started

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 Blackboard, in the Lab Materials folder.

Download LabWeek4examples.txt

We will also be using this stand-alone Python program and reading from these two text files.

Download processtextfile.py, desert.txt, CrimeAndPunishment.txt and Smart.English.stop.

Save them all in a folder where you keep materials for this class. As usual, start by:

```
>>> import nltk
```

Reading Text from a file

So far, we have used a small number of books from the Gutenberg Project that were provided in the NLTK. There is a larger collection available online, and you can see what is available in

http://www.gutenberg.org/catalog/

Text number 2554 in the catalog is the book "Crime and Punishment" by Dostoevsky, and we can access this text directly from Python by using url libraries to connect to the url and to read the content. This process is described in the NLKT book Chapter 3.

```
>>> from urllib import urlopen
>>> url = "http://www.gutenberg.org/files/2554/2554.txt"
>>> raw = urlopen(url).read()
>>> type(raw)
<type 'str'>
>>> len(raw)
1176893
>>> raw[:200]
```

Note that the text obtained from this web page is pretty clean, that is, it does not contain html tags or other web formatting that would interfere with text processing. So from the raw text, we could proceed with tokenization and further steps of text processing.

But if there is html formatting, we may have to take further steps to clean up the text. If you want to collect html formatted text, you may want to look at an example given in the NLTK book: http://news.bbc.co.uk/2/hi/health/2284783.stm

```
>>> blondurl = "http://news.bbc.co.uk/2/hi/health/2284783.stm"
>>> html = urlopen(blondurl).read()
>>> html[:1000]
```

Note that there is a lot of tags and formatting that is not text. One simple thing is to use the NLTK function clean_html to just remove html tags. If we wanted to do more sophisticated clean-up, we could investigate the python library BeautifulSoup.

```
>>> braw = nltk.clean_html(html)
>>> btokens = nltk.word_tokenize(braw)
>>> tbokens[:100]
```

[In NLTK 3.x, using the nltk.clean_html functions raises the following error: NotImplementedError ("To remove HTML markup, use BeautifulSoup's get_text() function")]

continue to cleanup the tokens to just get the text with no ads or other links

It's actually probably easiest to just collect text "by hand" and put it into a file and read this file. To see this, I have collected the text of a news stories by copying it directly from the web page and pasting it into a text file: desert.txt
I also went directly to the link for the Crime and Punishment book at the Gutenberg site and clicked on the link to the .txt file. From there I copy/pasted the main text into a text file and saved it as: CrimeAndPunishment.txt

Now reading a file is very simple in Python and the only difficulty is making sure that Python can find the file in the right directory. One option is to put the file that you want to read in the current directory of IDLE. To find out what that directory is, we can do:

```
>>> import os
>>> os.getcwd()
```

Then we could put desert.txt in that directory and just open it:

```
>>> f = open('desert.txt')
>>> text = f.read()
```

Or we can just read the file by giving the full path. Suppose that you added the text file to a directory on the C: drive called NLPclass\LabExamplesWeek3. Here you must substitute the path to the file desert.txt on your own lab machine.

```
>>> f = open('C:\\NLPclass\LabExamplesWeek3\desert.txt')
```

```
>>> text = f.read()
```

For Mac users, the path will use forward slash. For example, this is the path that I use on my mac laptop:

```
>>> f = open('/Users/njmccrac1/AAAdocs/NLPspring2015/labs/LabExamplesWeek3/desert.txt')
>>> text = f.read()
```

Stemming and Lemmatization

For this part, we will use the crime and punishment text from the file. Using one of the methods above, read the CrimeAndPunishment file.

```
>>> f = open('CrimeAndPunishment.txt')
>>> crimetext = f.read()
```

Tokenize the text and make crimewords to have lower-case words with no capitalization.

```
>>> crimetokens = nltk.word_tokenize(crimetext)
>>> len(crimetokens)
>>> crimetokens[:100]
```

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()
>>> lancaster = nltk.LancasterStemmer()
```

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

```
>>> crimePstem = [porter.stem(t) for t in crimetokens]
>>> crimePstem[:200]

>>> crimeLstem = [lancaster.stem(t) for t in crimetokens]
>>> crimeLstem[:200]
```

Note that the Lancaster stemmer has already lower-cased all the words, and appears to be a little more severe in removing word endings.

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()
>>> crimeLemma = [wnl.lemmatize(t) for t in crimetokens]
>>> crimeLemma[:200]
```

Note that the WordNetLemmatizer does not stem verbs and in general, doesn't stem very severely at all.

To see the results of more sentences stemmed and lemmatized in the NLTK, you can go to this NLKT stemmer and lemmatization demo page by Jacob Perkins:

http://text-processing.com/demo/stem/

Processing Text from Files – More about Mutual Information Frequencies

Open the processtextfile.py and read through it to observe how it reads text from the file CrimeAndPunishment.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 runs the bigram finder and scorers.

Modify the code to try the mutual information scores with and without the frequency filter. The Church and Hanks paper recommends using only frequencies 5 and above, but you may experiment with this.

Note that the Smart.English.stop word list does not match up with the tokens in the NLTK tokenizer because it is not removing tokens like "n't" that are produced by nltk.word_tokenize(). (By the way, note that there is a specialized Gutenberg stop word list that can be found online.)

For your homework, it is recommended that you develop a file similar to the processtextfile that processes both the documents of your analysis.

Lab Exercise

There is no work for the lab exercise for today; just post on the Lab Discussion what documents you are planning to process for your homework.