

NLP Lab Session Week 3  
September 14, 2017

## **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 Python interpreter in Jupyter notebook. As before, there are two versions of the python code used in this lab: a jupyter notebook and a python file.

Download LabWeek3Examples.ipynb or LabWeek3Examples.py

Download processtextfile.py, desert.txt, CrimeAndPunishment.txt and Smart.English.stop. You may need to copy/paste the .txt files from the Blackboard window.

Save them all in a folder where you keep materials for this class. Open your command prompt or terminal window and use the cd command to change directory to your class materials folder. Type at the prompt:

```
$ jupyter notebook
```

As usual, start your nlp session 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 request
>>> url = "http://www.gutenberg.org/files/2554/2554-0.txt"
>>> response = request.urlopen(url)
>>> raw = response.read().decode('utf8')
>>> 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> We must use the python decode function to convert the text to a Unicode string in python, where utf8 is one of the Unicode options. (In Python3, all strings are Unicode.)

```
>>> blondurl = "http://news.bbc.co.uk/2/hi/health/2284783.stm"
>>> html = request.urlopen(blondurl).read().decode('utf8')
>>> html[:1000]
```

Note that there are a lot of tags and formatting that is not text. We can use a Python library called BeautifulSoup to remove html markups. You must first import the package.

```
>>> from bs4 import BeautifulSoup
```

Get the contents as a BeautifulSoup object:

```
>>> soup = BeautifulSoup(html, 'lxml')
```

Use the get\_text function to get the contents of all the text tags.

```
>>> brow = soup.get_text()
>>> btokens = nltk.word_tokenize(brow)
>>> tbokens[:100]
```

But note that we would really like to get just the news text and not the extra links and ads. This can be done by using BeautifulSoup functions to follow the tree of tags and get the one we want. Documentation is found in: <https://www.crummy.com/software/BeautifulSoup/bs4/doc/> .

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.

[Optional for those who want to understand about the python environment: One option is to put the file that you want to read in the current directory of the Python interpreter. 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()
```

End Optional]

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\LabExamplesWeek4. Here you must substitute the path to the file desert.txt on your own computer.

[Note: if Jupyter Notebook is opened in this directory, then you can leave out the full path.]

```
>>> fin = open('C:\\NLPclass\\LabExamplesWeek4\\desert.txt')
```

```
>>> rawtext = fin.read()
```

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

```
>>> fin = open('/Users/njmccrac1/AAAdocs/NLPfall2017/labs/LabExamplesWeek3/desert.txt')
```

```
>>> rawtext = fin.read()
```

Now you can tokenize the text and continue with processing. As an example, we look at the KWIC of the word 'path'. In order to use the text concordance function, we first convert the list of tokens to an object of type nltk.Text. (Not all text processing requires this conversion.) The concordance and similarity functions are described in Chapter 1 of the NLTK book.

```
>>> deserttokens = nltk.word_tokenize(rawtext)
```

```
>>> text = nltk.Text(deserttokens)
```

```
>>> text.concordance('pass')
```

When we are done, we close the file.

```
>>> fin.close()
```

## **Stemming and Lemmatization**

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

```
>>> f = open('<put full path>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 two stemmers work on a small portion of the tokens.

```
>>> 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 lower-cased all the words, and in some cases, it appears to be a little more severe in removing word endings, but in others not.

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 NLTK stemmer and lemmatization demo page by Jacob Perkins:

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

## Processing Text from Files

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.

Now you can run a separate Python file by opening it in Jupyter or you can open another command prompt or terminal window and at the prompt run python:

```
$ python processtextfile.py
```

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**

1. First use `nltk.word_tokenize()` to find the tokens of `desert.txt`. Use NLTK’s Porter stemmer and Lancaster stemmer to stem the tokens of the `desert.txt` file. Choose a number randomly between 0 and 1363 (the length of the tokens is 1364). Post on the course discussion forum the word from `desert.txt` at that location from both the Porter and Lancaster stemmed token lists. Observe whether there was no stemming on that token, the stemming is the same or the stemming is different between the 2 stemmed lists.

2. Also post on the Lab Discussion what documents you are planning to process for your homework.