Exploring NLTK and Matplotlib in Python

1. NLTK (Natural Language Toolkit) and matplotlib are two packages widely used in python programming in the area of natural language processing. They allow scholars to conduct linguistic analyses (NLTK) and plot their results (matplotlib) with relative ease. In order to use python packages in your scripts, you must import them at the top of your document. Copy the following lines into the top of your script.

import matplotlib.pyplot as plt

import nltk

from nltk.tokenize import sent\_tokenize

from nltk.tokenize import word\_tokenize

from nltk import ne\_chunk

from nltk.corpus import stopwords

from nltk.util import ngrams

1. Now let’s bring in a text on which to conduct our analysis. Go to Project Gutenberg ([gutenberg.org](http://gutenberg.org)), and chose a literary text for your script. Download it as a flat file (.txt) and save it into a folder. Save your script into the same folder. Now add the following line to your script, replacing the **name of your file**:

text = open( '**nameofyourfile.**txt', mode='r', encoding='utf8').read()

1. In natural language processing, *tokenization* is the process of splitting your text into discrete sections for analysis. NLTK has functions that make this easy.

Tokenize your text into sentences, by using the function sent\_tokenize()

sentence\_tokens = sent\_tokenize(text)

Print sentence\_tokens and take a look at your result in the command line.

Now tokenize your text into words using NLTK’s word\_tokenize() function

word\_tokens = work\_tokenize(text)

Print out your results.

1. When conduction a natural language processing analysis, not all of the words in a given text may be important for your work. This is where stopwords come in useful. Stopwords in natural language processing are statistically highly common words, (such as ‘the’,’a’,’and’,’with’, and ‘for’ in English). NLTK has lists of stopwords in many different languages, and we can apply stopwords to our tokens in the following way:

stoplist = stopwords.words('english')

new\_word\_tokens = [word for word in new\_word\_tokens if word not in stoplist]

Note the second line is a list comprehension, which provides an easier way to create a new list than using a for loop.

1. NLTK also provides us a function to tag for parts of speech

partsofspeech = nltk.pos\_tag(new\_word\_tokens)

Print out partsofspeech and examine the results in your commandline

1. Named entity recognition (NER) is a process available in NLTK that classifies proper nouns. In order to perform this in NLTK, the word\_tokenize and pos\_tag functions must be done first, as we have. Add the following line to your script:

named\_entities\_included = ne\_chunk(partsofspeech)

Print this out and take a look in your command line.

1. N-grams. Beyond creating lists of individual words, NLTK can provide us with lists of n-gram, words that appear next to each other. Print the bigrams and trigrams in your text with the help of the following lines:

bigrams = list(ngrams(new\_word\_tokens,2))

trigrams = list(ngrams(new\_word\_tokens,3))

1. Frequency distribution and rare words. NLTK can show us which words are frequent and infrequent in a given text. Print the 50 rarest words and 50 most common words in your text.

freq\_dist = nltk.FreqDist(new\_word\_tokens)

rarewords = list(freq\_dist.keys())[-50:]

commonwords = freq\_dist.most\_common(50)

1. Plotting findings with matplotlib

lengthsarray = []

for x in sentence\_tokens:

length = len(x)

lengthsarray.append(length)

Let’s look at the new list.

print (lengthsarray)

We can plot the distribution of these sentence length using a histogram with matplotlib:

plt.hist(lengthsarray)

To see this we need to add the line:

plt.show()

10) Now we have a graphic that shows the distribution, but we’re missing labels. Let’s add some labels, a title, a label for the x-axis, and a label for the y-axis.

xaxislabel = 'Number of characters in sentence.'

yaxislabel = 'Frequency'

title = 'Sentence Length in \_\_\_\_\_\_\_’

plt.xlabel(xaxislabel)

plt.ylabel(yaxislabel)

plt.title(title)

You’ve now had a small taste of what is possible with NLTK and matplotlib. This is barely scratching the surface, as these packages have a wide array of functionality, and there are many books and tutorials written about both. If you’re interested in learning more about natural language processing in python, you’ll definitely want to explore NLTK and matplotlib further! Take a look at the “Resources” handout for information on useful places to find more information on python natural language processing topics relating to digital humanities.