## Lab 1 – Find Frequency Distributions with NLTK (1)

Today we will be learning how to use the Natural Language Toolkit (NLTK) module to perform basic text analysis.

NLTK is a Python add-on package which has multiple useful functions for analyzing text files. One key component of text analysis is finding keyword frequency distributions.

A frequency distribution can be shown as a table containing how many times within a document different words appear. To directly quote the NLTK book, “it’s called a "distribution" because it tells us how the total number of word tokens in the text are distributed across the vocabulary items. Since we often need frequency distributions in language processing, NLTK provides built-in support for them” (<http://www.nltk.org/book/ch01.html>).

Under blackboard, you can find the following documents:

* Installation instruction
* Lab1.doc (tutorial that you can go over by yourself after the lab)
* Lab1.ipynb (material that will be present during the lab)
* Frequency.py (code to create frequency distribution)
* Frequency without stopwords.py (code to create frequency distribution without stop words)
* Input.txt

*If you want to follow the instructor during the lab, please get your jupyter notebook ready before the lab. The installation instruction has been posted on the Blackboard.*

1. **Read data**

f = open('input.txt','rU') #read file from local

raw = f.read()

raw = raw.replace('\n',' ') #replace all ‘\n’ by ‘ ’(space)

raw = raw.decode('utf8') #decode raw text by utf-8

1. **Tokenization**

#here we use the built-in NLTK tokenization function to tokenize

tokens = nltk.word\_tokenize(raw)

#change all tokens into lower case

#option 1: list comprehension

words1 = [w.lower() for w in tokens]

#option 2:

words1 = []

for w in tokens:

words1.append(w.lower())

#only keep text words, no numbers

words2 = [w for w in words1 if w.isalpha()]

#encode the raw token list by utf-8

words3 = [w.encode('utf8') for w in words2]

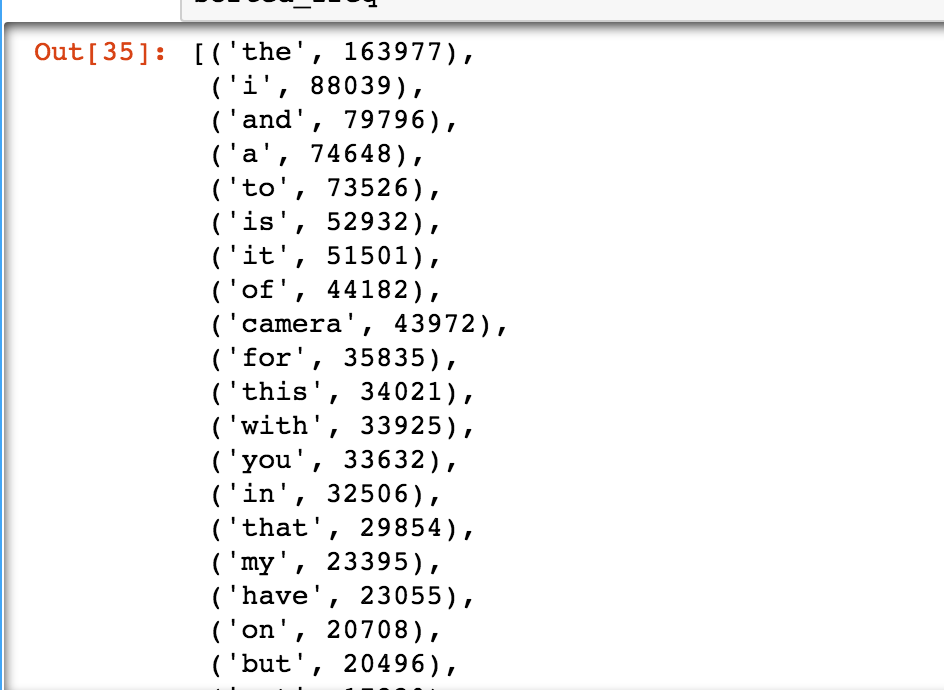
#generate a frequency dictionary for all tokens

freq = FreqDist(words3)

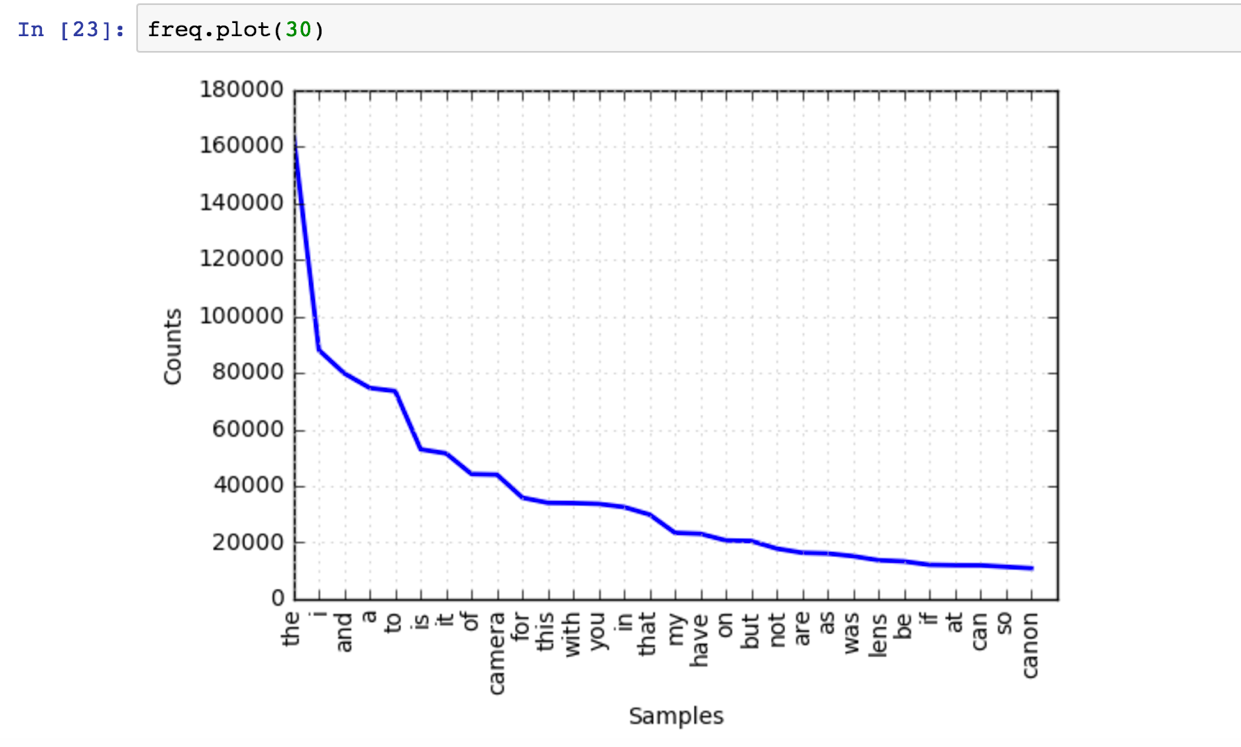
#sort the frequency list in descending order

sorted\_freq = sorted(freq.items(),key = lambda k:k[1], reverse = True)

Top Frequency words list with their word-frequency



Top Frequency words distribution graph



Interpretation:

The top frequency words are all meaningless words, such as ‘the’, ‘i’, ‘and’, etc, which are useless if we want to understand the input text data. Therefore, the stop words removal step is very important.

1. **Stop words removal**

#the following steps keep the words which are not in the stop words list

#here we use the built-in NLTK English stop words list

from nltk.corpus import stopwords

stopwords = stopwords.words('english')

#only keep the words that not in nltk stopwords word list

words\_nostopwords = [w.encode('utf8') for w in words2 if w not in stopwords]

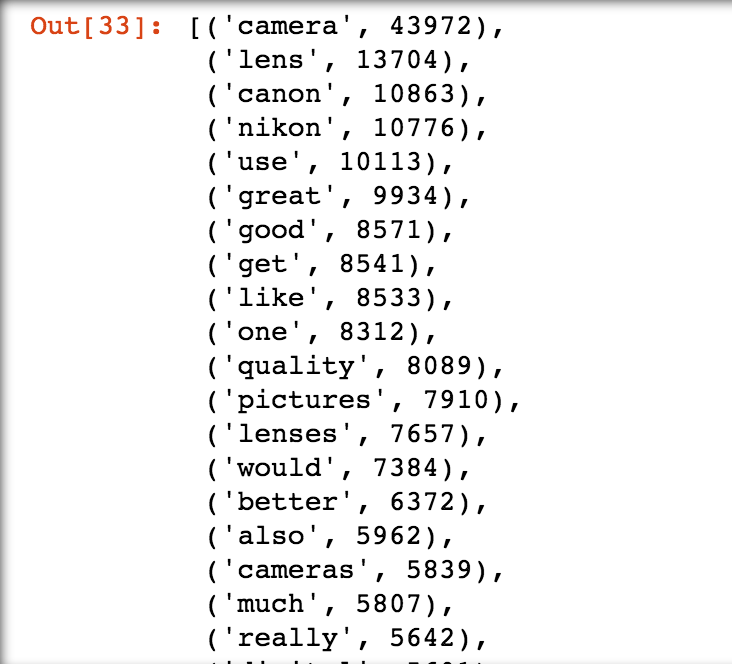
#generate a frequency dictionary for all tokens

freq\_nostw = FreqDist(words\_nostopwords)

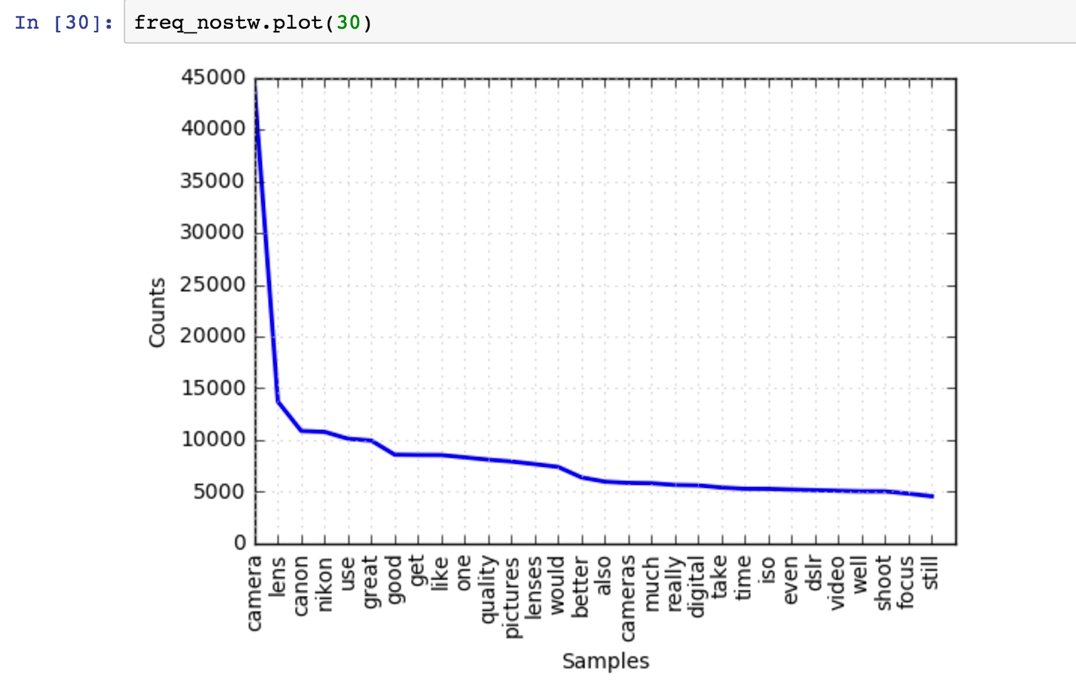
#sort the frequency list in decending order

sorted\_freq\_nostw = sorted(freq\_nostw.items(),key = lambda k:k[1], reverse = True)

Top frequency words without stop words



Top frequency words without stop words distribution graph



Interpretation:

After removing the stop words, the result makes more sense. Top frequency words are: “camera”, “lens”, “canon”, “nikon”, etc, which apparently indicates the input text is an article about camera. Stop words removal helps to remove meaningless words, and to bring valuable information in front of people.

1. **Save the result**

with open ('output.txt','a') as outfile:

for line in sorted\_freq\_nostw:

outfile.write(str(line[0])+'\t'+str(line[1])+'\n')