PYTHON TEXT MINING HOMEWORK 3

By – Syed Zeeshan Ali

M10614655 | [alisz@mail.uc.edu](mailto:alisz@mail.uc.edu)

**18. Read in some text from a corpus, tokenize it, and print the list of all *wh*-word types that occur. (*wh*-words in English are used in questions, relative clauses and exclamations: *who*, *which*, *what*, and so on.) Print them in order. Are any words duplicated in this list, because of the presence of case distinctions or punctuation?**

from nltk.corpus import gutenberg #import gutenberg

import nltk #

import re

fd=nltk.FreqDist([w for w in gutenberg.words('austen-emma.txt') if re.search('^[Ww]h.\*',w) ])

#Reading austen-emma.txt word by word

#The regex used in this case is ‘^[Ww]h.\*’ it checks whether a word starts with capital or small w,

#followed by h and then o or more single characters.

#FreqDist function will give us the frequency distribution

for m in fd:

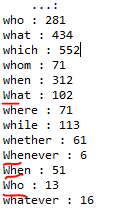
print(m,":",fd[m])

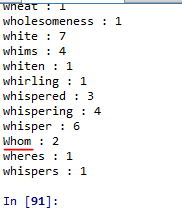
#Plotting the frequency distribution of words

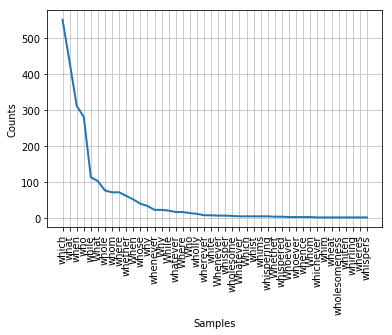
fd.plot()

#The regex used in this case is ‘^[Ww]h.\*’ it checks whether a word starts with capital or small w, #followed by h and then o or more single characters. We can see there are **repetition in basic words #such as What, Which, When, Who etc. because of case** difference in w.

**OUTPUT:**



.  
.  




**Fig Displaying Frequency Distribution of words starting wh**

**19. Create a file consisting of words and (made up) frequencies, where each line consists of a word, the space character, and a positive integer, e.g. fuzzy 53. Read the file into a Python list using open(filename).readlines(). Next, break each line into its two fields using split(), and convert the number into an integer using int(). The result should be a list of the form: [['fuzzy', 53], ...]**

**#19.**

import os #import os package

os.chdir("C:\Study\Python") #Setting the current working directory

target = open("question19.txt", 'w')

#Opening the file in write mode, filehandle used is target

target.write("Zeeshan 45")

target.write("\n") #Writing new line character

target.write("Ali 77")

target.write("\n")

target.write("Aloo 90")

target.write("\n")

target.write("baingan 22")

target.write("\n")

#Writing lines in file

#Closing the file handle

target.close()

#Reading lines of file

red=open("question19.txt").readlines()

a=[] #declaring empty list

for line in red:

a.append((line.split(" ")))

#Appending elements in empty list after splitting on " "

#Replacing \n and converting into int

a[0][1]=int(re.sub(r"\n","",a[0][1]))

a[1][1]=int(re.sub(r"\n","",a[1][1]))

a[2][1]=int(re.sub(r"\n","",a[2][1]))

a[3][1]=int(re.sub(r"\n","",a[3][1]))

#Printing the final list

print(a)

**OUTPUT:**



21. ◑ **Write a function unknown() that takes a URL as its argument, and returns a list of unknown words that occur on that webpage. In order to do this, extract all substrings consisting of lowercase letters (using re.findall()) and remove any items from this set that occur in the Words Corpus (nltk.corpus.words). Try to categorize these words manually and discuss your findings.**

#21.

from urllib import request

from bs4 import BeautifulSoup

import re

from nltk.corpus import words

def unknown(url):

#Response from web page

page= request.urlopen(url).read().decode('utf8') #read and decode UTF-8 format

bs = BeautifulSoup(page).get\_text() #beautiful soup for web scraping

corpWords = set(words.words()) #Corpus Dictionary words

pg\_words = re.findall(r'\b\w+', bs) #Find all words starting with a boundary

uniqueWords=[word for word in pg\_words if word not in corpWords] #iterate through all html words

#Finding words which are not present in corpus word dictionary

print(set(uniqueWords)) #Print unique words from html that are not in dict

url1="http://www.nltk.org/"

unknown(url1) #Calling function with argument

**OUTPUT:**

**{'FILE\_SUFFIX', 'At', 'using', 'guides', 'NLTK', 'Index', 'categorizing', 'Next', 'word\_tokenize', 'CD', 'Copyright', 'Bird', '2015', 'OS', 'Thanks', 'Module', 'topics', 'Of', 'Project', 'libraries', 'tagging', 'COLLAPSE\_INDEX', 'Windows', 'Best', 'ne\_chunk', 'modules', 'pos\_tag', 'Identify', 'linguists', '2', 'WordNet', 'Display', 'http', 'called', 'Table', 'Contribute', 'NNP', 'SOURCELINK\_SUFFIX', '2009', 'Source', 'hands', 'Tokenize', 'programs', 'Sphinx', 'students', 'educators', 'creators', 'provides', 'Inc', '50', 'announcements', 'parsing', 'HOWTO', 'VERSION', 'Linux', 'txt', 'Loper', 'html', 'book\_1ed', 'Python', 'has', 'named', '3', 'Processing', 'parsed\_sents', 'mrg', 'Reilly', 'tokens', '6', 'analyzing', 'VB', 'API', 'Last', '02', 'Toolkit', 'follows', 'researchers', 'Language', 'RB', '0', 'Search', 'Tree', 'NLP', '5', 'IN', 'Created', 'Some', 'wrappers', 'JJ', 'Klein', 'engineers', 'fundamentals', 'things', 'updated', 'nltk', 'Natural', 'If', 'uses', 'NN', 'Written', 'interfaces', 'NB', 'Show', 'HAS\_SOURCE', 'users', 'var', 'entities', 'Data', '1', 'Ewan', 'DOCUMENTATION\_OPTIONS', 'Installing', 'FAQ', 'wsj\_0001', 'It', 'VBD', 'URL\_ROOT', 'tokenization', 'resources', 'Wiki', '2017', 'org', 'didn', 'PERSON', 'processing', 'treebank', 'News', 'Contents', 'Steps', 'introducing', 'programming'}**

**23.**  **Are you able to write a regular expression to tokenize text in such a way that the word *don't* is tokenized into *do* and *n't*? Explain why this regular expression won't work: «n't|\w+».**

#23.

import nltk

import re

import os

# opening text file containing don't

os.chdir("C:\Study\Python")

f = open('trial.txt','w')

f.write("don't don't don't don't")

f.close()

f1 = open('trial.txt','r')

pal=f1.read()

#reading file

f1.close()

#regex to get groups of do and n't

zz = re.findall(r'\b(do)(n\'t)', pal)

print(zz)

**OUTPUT:**



**24.** **Try to write code to convert text into hAck3r, using regular expressions and substitution, where e → 3, i → 1, o → 0, l → |, s → 5, . → 5w33t!, ate → 8. Normalize the text to lowercase before converting it. Add more substitutions of your own. Now try to map s to two different values: $ for word-initial s, and 5 for word-internal s.**

#24.

import nltk

import re

# reads in a text

f1 = "Sample noob for becoming a hacker. 1 for the team, i love you let's rock YOLO Supra"

# lowercases the text

f1 = f1.lower()

def convertHacker(text):

"""conversion hAck3r"""

new\_text = []

#initial pass subsitutes 8 for ate.

z = re.compile(r'ate')

text = z.sub('8', text)

# regex that searches through the text to find instances of the letters to be converted.

z = re.compile(r'[eiols]|\.')

# converts all the letters

for w in text:

if re.search(z, w):

if w == 'e':

w = '3'

elif w == 'i':

w = '1'

elif w == 'o':

w = '0'

elif w == 's':

w = '5'

elif w == 'l':

w = '|'

elif w == '.':

w = '5w33t!'

new\_text.extend(w)

new\_text = ''.join(new\_text)

# regex searching for word initial s.

z = re.compile(r'\b5')

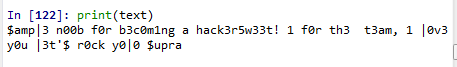
ntext = z.sub('$', new\_text)

return ntext

text = convertHacker(f1)

print(text)

**OUTPUT:**



**27. Python's random module includes a function choice() which randomly chooses an item from a sequence, e.g. choice("aehh ") will produce one of four possible characters, with the letter h being twice as frequent as the others. Write a generator expression that produces a sequence of 500 randomly chosen letters drawn from the string "aehh ", and put this expression inside a call to the ''.join()function, to concatenate  them into one long string. You should get a result that looks like uncontrolled sneezing or maniacal laughter: he haha ee heheeh eha. Use split() and join() again to normalize the whitespace in this string**

#27.

from random import choice

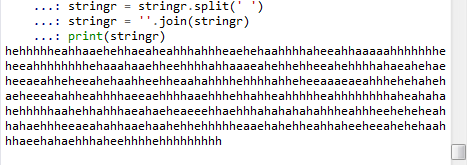
stringr = ''.join(list((choice("aehh ") for x in range(500))))

stringr = stringr.split(' ')

stringr = ''.join(stringr)

print(stringr)

**OUTPUT:**



**31. Define the variable saying to contain the list ['After', 'all', 'is', 'said', 'and', 'done', ',', 'more','is', 'said', 'than', 'done', '.']. Process this list using a for loop, and store the length of each word in a new list lengths. Hint: begin by assigning the empty list to lengths, using lengths = []. Then each time through the loop, use append() to add another length value to the list. Now do the same thing using a list comprehension.**

#31.

saying = ['After', 'all', 'is', 'said', 'and', 'done', ',', 'more', 'is', 'said', 'than', 'done', '.']

length1 = [] #Declaring empty list

for i in saying:

length1.append(len(i)) #Appending in list from original list

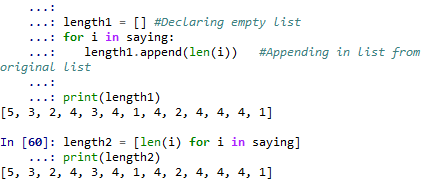
print(length1)

# list comprehension

length2 = [len(i) for i in saying]

print(length2)

**OUTPUT:**



**32.** Define a variable silly to contain the string: 'newly formed bland ideas are inexpressible in an infuriating

way'. (This happens to be the legitimate interpretation that bilingual English-Spanish speakers can assign to Chomsky's famous nonsense phrase, *colorless green ideas sleep furiously* according to Wikipedia). Now write code to perform the following tasks:

1. Split silly into a list of strings, one per word, using Python's split() operation, and save this to a variable called bland.
2. Extract the second letter of each word in silly and join them into a string, to get 'eoldrnnnna'.
3. Combine the words in bland back into a single string, using join(). Make sure the words in the resulting string are separated with whitespace.
4. Print the words of silly in alphabetical order, one per line.

#32.

silly = 'newly formed bland ideas are inexpressible in an infuriating way'

#Declaring silly string

#a. Splitting silly

bland = silly.split(' ')

#Empty string

silly\_word = ''

#b. Printing eoldrnnnna

for word in bland:

silly\_word += word[1]

print(silly\_word)

#c.

new\_bland = ' '.join(bland)

print(new\_bland)

print(bland)

#d. silly in alphabetical order, one per line.

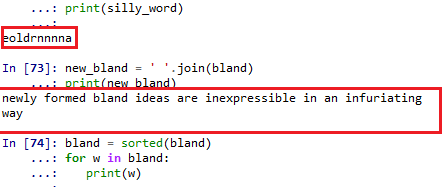
bland = sorted(bland)

for w in bland:

print(w)

**OUTPUT:**



****

