HU Extension Assignment 12 E63 Big Data Analytics

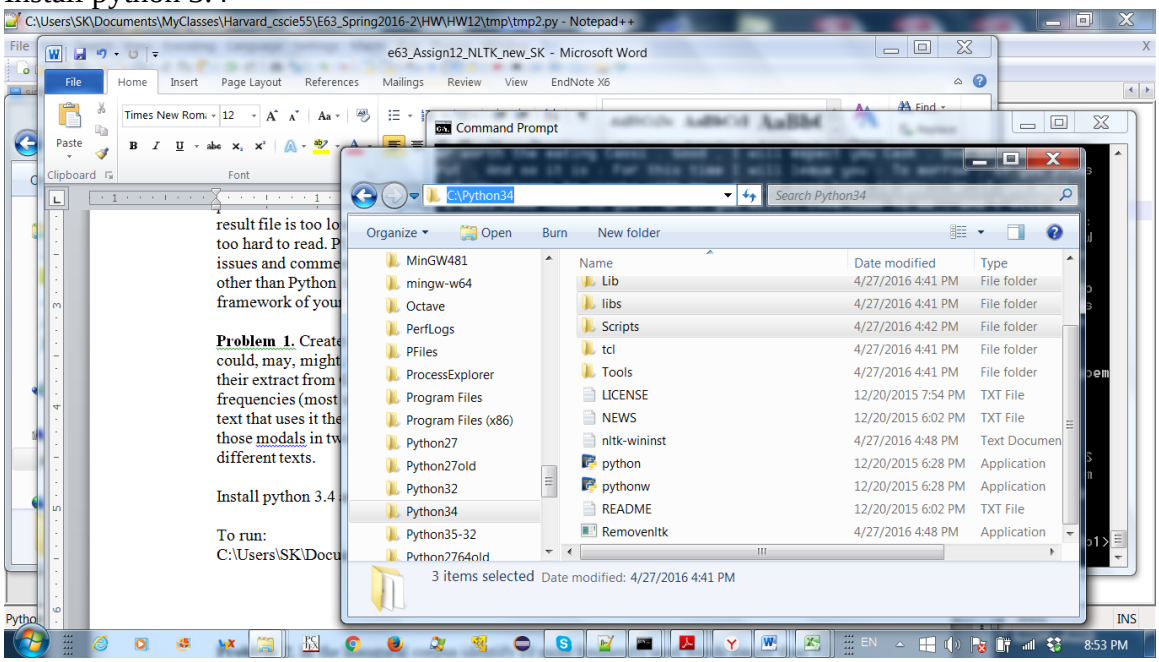
Handed out: 04/23/2016 Due by 11:30PM EST on Friday, 04/29/2016
Solution: Serguey Khovansky

Please, describe every step of your work and present all intermediate and final results in a Word document. Please, copy past text version of all essential command and snippets of results into the Word document with explanations of the purpose of those commands. We cannot retype text that is in JPG images. Please, always submit a separate copy of the original, working scripts and/or class files you used. Sometimes we need to run your code and retyping is too costly. Please include in your MS Word document only relevant portions of the console output or output files. Sometime either console output or the result file is too long and including it into the MS Word document makes that document too hard to read. PLEASE DO NOT EMBED files into your MS Word document. For issues and comments visit the class Discussion Board. If you use some other language other than Python in your daily work with NLP, please be free to use that language and a framework of your choice to do this assignment.

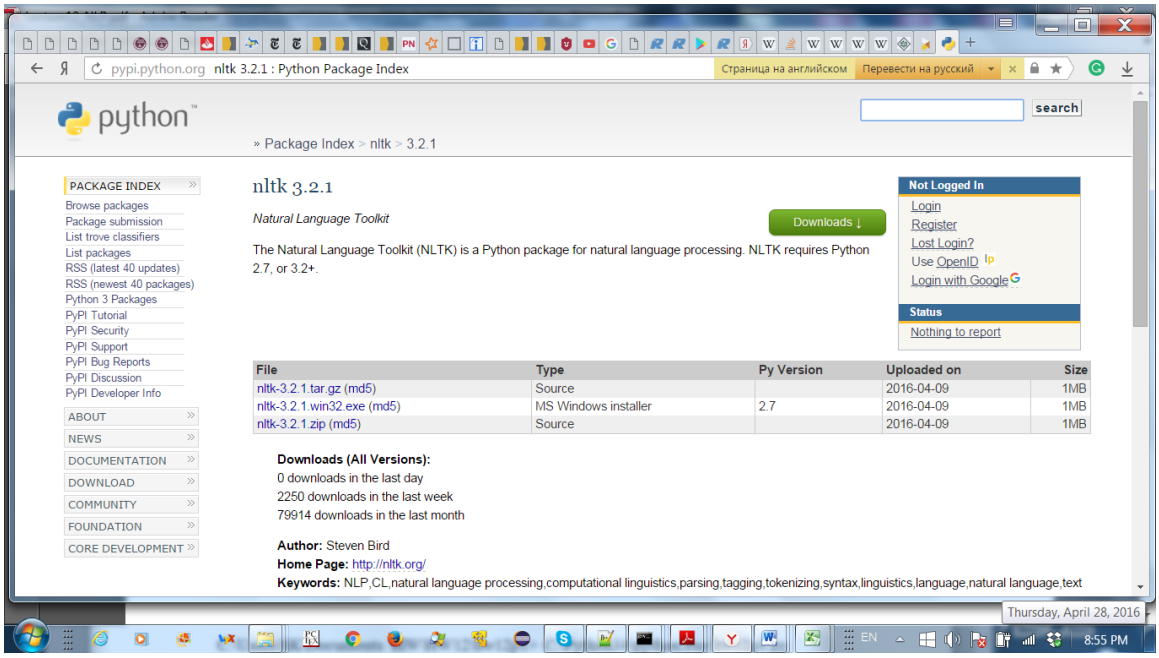
Problem 1. Create a table displaying **relative** frequencies with which "modals" (can, could, may, might, will, would and should) are used in 18 texts provided by NLTK in their extract from Gutenberg Corpus. For two modals with the largest span of relative frequencies (most used minus least used), select a text which uses it the most and the text that uses it the least. Compare usage in both texts by examining the concordances of those modals in two texts. Perhaps try to understand how are those words used in different texts.

Solution

Install python 3.4



Install nltk



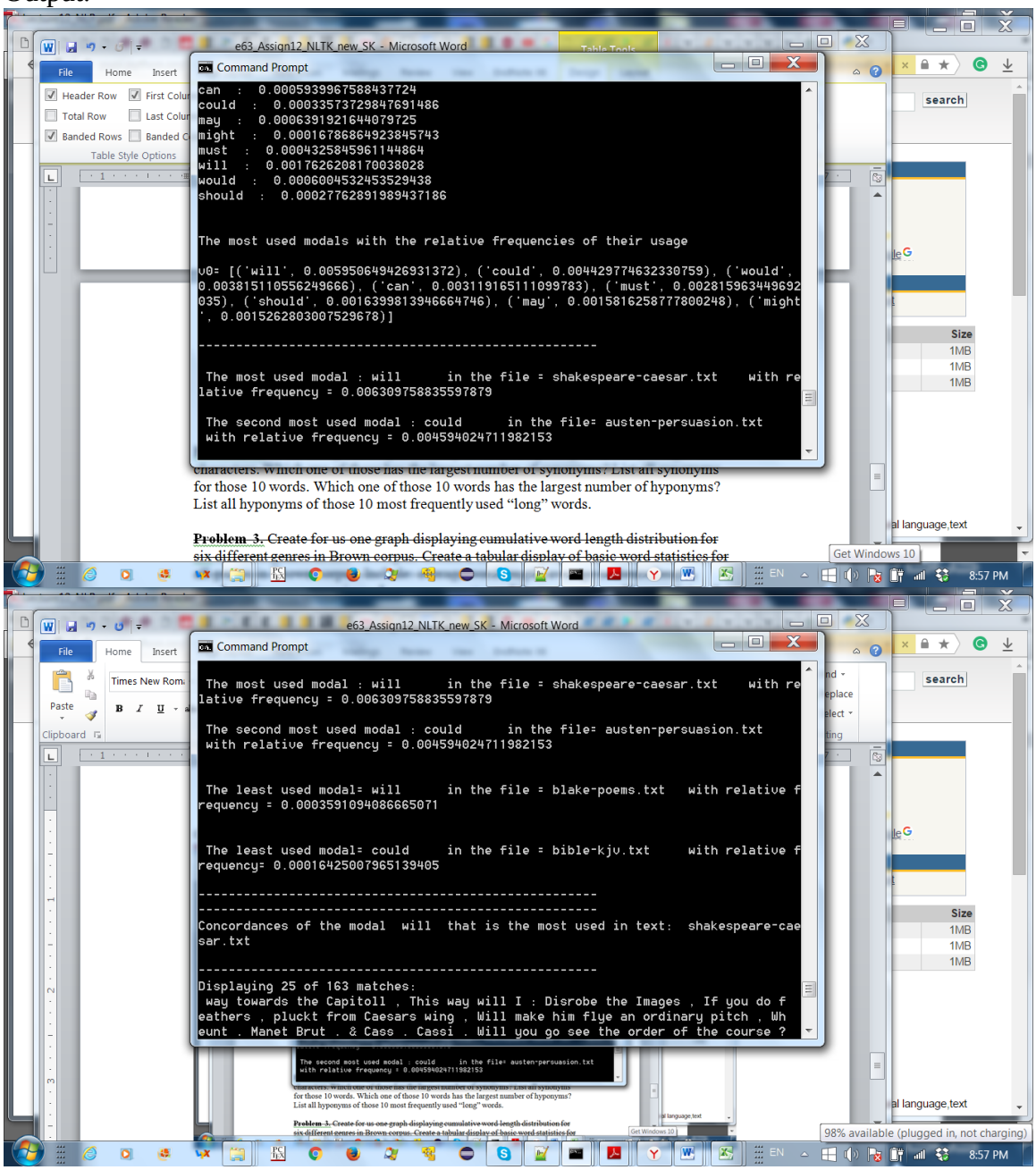
The code for the problem 1 (it can also can be found in the attached file hw12p1.py):

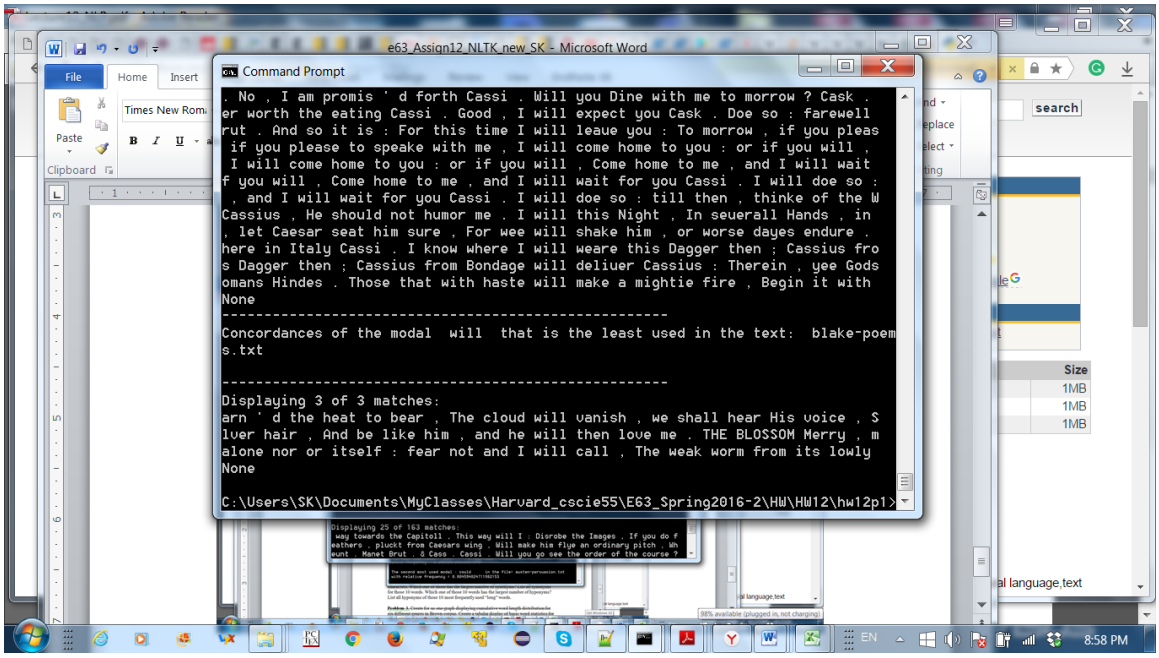
```
import operator
import nltk
from nltk.book import gutenberg

modals=['can', 'could', 'may', 'might', 'must', 'will', 'would', 'should']

# create dictionary
d={}
# fill dictionary with relative frequencies
for fileid in gutenberg.fileids():
    d[fileid]={}
    fdist = nltk.FreqDist(w.lower() for w in nltk.corpus.gutenberg.words(fileid))
    total_words=len(nltk.corpus.gutenberg.words(fileid))
    for m in modals:
#        d[fileid][m] = fdist[m]
        d[fileid][m] = fdist[m]/total_words

# print dictionary
print('\n\n ------ PRINT DICTIONARY OF RELATIVE FREQUENCIES -------')
for fileid in gutenberg.fileids():
    print(fileid + ' ')
    for m in modals:
        print(m,' : ', d[fileid][m] ,' ')
    print('\n')

# find the most and least used modals
d_most={}
d_name_most={}
d_least={}
d_name_least={}
for m in modals:
    d_most[m]=-1
    d_least[m]=1000000
    d_name_least[m]={}
    d_name_most[m]={}
    for fileid in  gutenberg.fileids():
        if d[fileid][m] > d_most[m]:
            d_most[m] = d[fileid][m]
            d_name_most[m] = fileid
        if d[fileid][m] < d_least[m]:
```

```
            d_least[m] = d[fileid][m]
            d_name_least[m] = fileid


#for m in modals:
#    print ('m=',m, ' d_most[m]=',d_most[m], '  d_name_most[m]=',d_name_most[m],'\n')
#print('\n')
#for m in modals:
#    print ('m=',m, ' d_least[m]=',d_least[m], ' d_name_least[m]=',d_name_least[m],'\n')

# compute spans
d_span={}
for m in modals:
   d_span[m]= d_most[m]-d_least[m]


max_span_m0 = max(d_span.items(), key=operator.itemgetter(1))[0]
max_span_m1 = max(d_span.items(), key=operator.itemgetter(1))[1]

#sort the spans
v0=[k for k in sorted(d_span.items(), key=operator.itemgetter(1), reverse=True)]

print('The most used modals with the relative frequencies of their usage', '\n')
print('v0=',v0,'\n')

#select a text which uses it the most
d_name_most=''
def textname_mostusedmodal(modal_ans1):
    imax=-1
    for fileid in gutenberg.fileids():
        if d[fileid][modal_ans1] > imax:
            imax = d[fileid][modal_ans1]
            d_name_most = fileid
    return     d_name_most, imax

#select a text which uses it the least
def textname_leastusedmodal(modal_ans1):
    imin = 10000000
    for fileid in gutenberg.fileids():
        if d[fileid][modal_ans1] < imin:
            imin = d[fileid][modal_ans1]
            d_name_most = fileid
    return     d_name_most, imin

modal_ans1=v0[0][0]
d_name_most1, imax1 = textname_mostusedmodal(modal_ans1)

modal_ans2=v0[1][0]
d_name_most2, imax2 = textname_mostusedmodal(modal_ans2)

leastUsed1, imin1 = textname_leastusedmodal(modal_ans1)
leastUsed2, imin2 = textname_leastusedmodal(modal_ans2)

print('-------------------------------------------------------')
print('\n The most used modal :', modal_ans1, '\t in the file =',d_name_most1,'\t with relative frequency =',imax1, '\n')
print(' The second most used modal :', modal_ans2, '\t in the file=',d_name_most2,'\t with relative frequency =',imax2, '\n')
print('\n The least used modal=', modal_ans1, '\t in the file =',leastUsed1,'\t with relative frequency =',imin1, '\n')
print('\n The least used modal=', modal_ans2, '\t in the file =',leastUsed2,'\t with relative frequency=',imin2, '\n')
print('-------------------------------------------------------')

print('-------------------------------------------------------')
print('Concordances of the modal ',modal_ans1, ' that is the most used in text: ',d_name_most1,'\n' )
print('-------------------------------------------------------')
print( nltk.Text(nltk.corpus.gutenberg.words(d_name_most1)).concordance(modal_ans1)  )

print('-------------------------------------------------------')
print('Concordances of the modal ',modal_ans1, ' that is the least used in the text: ',leastUsed1,'\n' )
print('-------------------------------------------------------')
print( nltk.Text(nltk.corpus.gutenberg.words(leastUsed1)).concordance(modal_ans1)  )
```

To run:
C:\Users\SK\Documents \HW\HW12\hw12p1> C:\Python34\python.exe hw12p1.py

Output:





Problem 2. In the Inaugural corpus identify 10 most frequently used words longer than 7 characters. Which one of those has the largest number of synonyms?

Relevant part of the code:

```
import operator
import nltk
from nltk.corpus import inaugural
from nltk.corpus import wordnet as wn


#identify frequency of words used in all texts related to inaugural
d={}
for fileid in inaugural.fileids():
    fdist = nltk.FreqDist(w.lower() for w in nltk.corpus.inaugural.words(fileid))
    d={k: d.get(k, 0) + fdist.get(k, 0) for k in set(d) | set(fdist)}

# filter keys of length in excess of 7
d = {k: v for k, v in d.items() if len(k) > 7}

# sort dictionary by values
v0=[k for k in sorted(d.items(), key=operator.itemgetter(1), reverse=True)]

#Take the first 10 items and place them into dictionary ( word: [synonyms]
dsyn={}
# dictionary for [word: length of synonyms]
dmaxNumSyn={}
for i in range(1,10):
#  get  Synset of a particular word 'i' held in v0[i][0]
    tmpv = wn.synsets(v0[i][0])
    len_syn = len(tmpv)
#                Get a set, so far empty, for synonymous
    set_synonims = set()
#   Fill the set with synonymous of lemma_names
    for k in range(0, len_syn-1):
        set_synonims.update(tmpv[k].lemma_names())
#Fill the dictionary dsyn
    dsyn[v0[i][0]]=set_synonims
    dmaxNumSyn[v0[i][0]]=len_syn
```

```
print('LIST ALL SYNONYMS FOR THOSE 10 WORDS')
for i in range(1,10):
   print(v0[i][0],' : ', dsyn[v0[i][0]] ,'   \n')

print('LIST ALL WORDS AND THE NUMBER OF THEIR  SYNONYMS')
print(dmaxNumSyn)

#sort the spans
dmaxsyn=[k for k in sorted(dmaxNumSyn.items(), key=operator.itemgetter(1), reverse=True)]

print('The word with maximum number of synonyms is  \n')
print(dmaxsyn[0])
```

Answers:

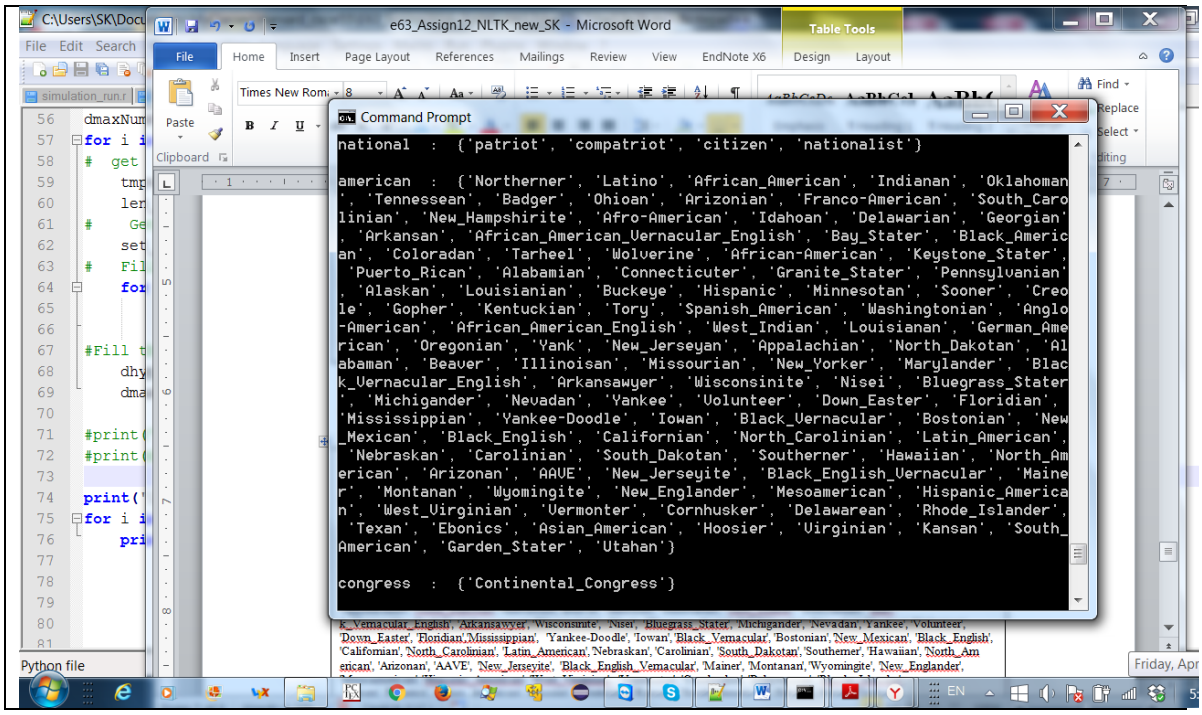
```
The word with maximum number of synonyms is

('constitution', 16)
```

List all synonyms for those 10 words.

```
LIST ALL SYNONYMS FOR THOSE 10 WORDS
citizens  :  set()

constitution  :  {'organic_law', 'establishment', 'organization', 'fundamental_law', 'makeup', 'U.S._Constitution',
'Constitution_of_the_United_States', 'United_States_Constitution', 'composition', 'organisation', 'formation', 'constitution
', 'Constitution', 'physical_composition', 'US_Constitution', 'make-up'}

national  :  {'national', 'home', 'subject', 'interior', 'internal'}

american  :  {'American_language', 'American_English', 'American'}

congress  :  {'Congress', 'congress', 'United_States_Congress', 'U.S._Congress',  'US_Congress'}

interests  :  {'concern', 'involvement', 'worry', 'interest_group', 'stake', 'sake', 'pastime', 'pursuit', 'occupy', 'interestingness', 'interest'}

political  :  {'political'}

executive  :  {'administrator', 'executive_director', 'executive'}

principles  :  {'principle', 'rule', 'precept'}
```

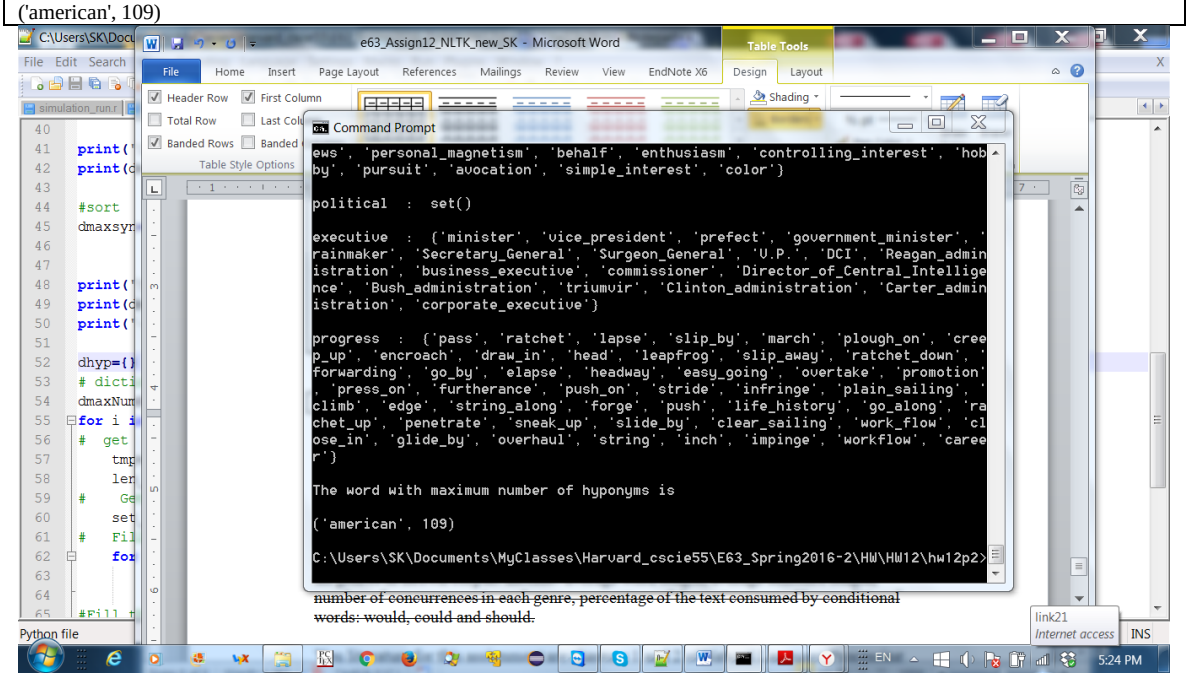
List all hyponyms of those 10 most frequently used “long” words.
Sample output (all output is too long)

```
constitution  :  {'communization', 'collectivization', 'karyotype', 'genotype', 'collectivisation', 'genetic_constitution', 'structure',
'colonisation', 'unionisation', 'texture', 'settlement', 'grain', 'colonization', 'communisation', 'unionization', 'phenotype', 'federation'}

national  :  {'patriot', 'compatriot', 'citizen', 'nationalist'}

american  :  {'Northerner', 'Latino', 'African_American', 'Indianan', 'Oklahoman', 'Tennessean', 'Badger', 'Ohioan', 'Arizonian', 'Franco-
American', 'South_Carolinian', 'New_Hampshirite', 'Afro-American', 'Idahoan', 'Delawarian', 'Georgian', 'Arkansan',
'African_American_Vernacular_English', 'Bay_Stater', 'Black_American', 'Coloradan', 'Tarheel', 'Wolverine', 'African-American',
'Keystone_Stater', 'Puerto_Rican', 'Alabamian', 'Connecticuter', 'Granite_Stater', 'Pennsylvanian', 'Alaskan', 'Louisianian', 'Buckeye',
'Hispanic', 'Minnesotan', 'Sooner', 'Creole', 'Gopher', 'Kentuckian', 'Tory', 'Spanish_American', 'Washingtonian', 'Anglo
-American', 'African_American_English', 'West_Indian', 'Louisianan', 'German_American', 'Oregonian', 'Yank', 'New_Jerseyan',
'Appalachian', 'North_Dakotan', 'Alabaman', 'Beaver', 'Illinoisan', 'Missourian', 'New_Yorker', 'Marylander', 'Blac
k_Vernacular_English', 'Arkansawyer', 'Wisconsinite', 'Nisei', 'Bluegrass_Stater', 'Michigander', 'Nevadan', 'Yankee', 'Volunteer',
'Down_Easter', 'Floridian','Mississippian', 'Yankee-Doodle', 'Iowan', 'Black_Vernacular', 'Bostonian', 'New_Mexican', 'Black_English',
'Californian', 'North_Carolinian', 'Latin_American', 'Nebraskan', 'Carolinian', 'South_Dakotan', 'Southerner', 'Hawaiian', 'North_Am
erican', 'Arizonan', 'AAVE', 'New_Jerseyite', 'Black_English_Vernacular', 'Mainer', 'Montanan', 'Wyomingite', 'New_Englander',
'Mesoamerican', 'Hispanic_American', 'West_Virginian', 'Vermonter', 'Cornhusker', 'Delawarean', 'Rhode_Islander',
 'Texan', 'Ebonics', 'Asian_American', 'Hoosier', 'Virginian', 'Kansan', 'South_American', 'Garden_Stater', 'Utahan'}
```



Which one of those 10 words has the largest number of hyponyms?

('american', 109)



Related part of the code:

```
dhyp={}
# dictionary for [word: length of synonyms]
dmaxNumHyp={}
for i in range(1,10):
#  get  Synset of a particular word 'i' held in v0[i][0]
   tmpv = wn.synsets(v0[i][0])
   len_syn = len(tmpv)
#              Get a set, so far empty, for hyponyms
```

```
    set_hyponyms = set()
#   Fill the set with hyponyms of lemma_names
    for k in range(0, len_syn-1):
        tmp=[lemma.name() for synset in tmpv[k].hyponyms() for lemma in synset.lemmas()]
        set_hyponyms.update(tmp)
#Fill the dictionary
    dhyp[v0[i][0]] = set_hyponyms
    dmaxNumHyp[v0[i][0]]=len(set_hyponyms)

print('LIST ALL HYPONYMS FOR THOSE 10 WORDS')
for i in range(1,10):
    print(v0[i][0],' : ', dhyp[v0[i][0]] ,'   \n')

#sort
dmaxhyp=[k for k in sorted(dmaxNumHyp.items(), key=operator.itemgetter(1), reverse=True)]

print('The word with maximum number of hyponyms is  \n')
print(dmaxhyp[0])
```

The entire code can be found in the file hw12p2.py

Problem 3. ~~Create for us one graph displaying cumulative word length distribution for six different genres in Brown corpus. Create a tabular display of basic word statistics for all genres in Brown corpus. Include: average word length, average sentence length, number of concurrences in each genre, percentage of the text consumed by conditional words: would, could and should.~~

You literature for this assignment are chapters 1 and 2 of Natural Language Processing with Python book by Steven Bird et al.