Web Mining LA1

August 8, 2020

1 NAME: SREEMANTH

2 REG. NO: 18BCE0745

2.1 Question 1

[80]:
```
import re
from bs4 import BeautifulSoup
from urllib import request
```

2.1.1 a. Extract the source content (excluding any tags) from the website (https://en.wikipedia.org/wiki/Web_mining).

[81]:
```
url = "https://en.wikipedia.org/wiki/Web_mining"
html = request.urlopen(url).read().decode('utf8')
raw = BeautifulSoup(html, 'html.parser').find('p').get_text()
```

[82]:
```
print(raw)
```

```
Web mining is the application of data mining techniques to discover patterns
from the World Wide Web. As the name proposes, this is information gathered by
mining the web. It makes utilization of automated apparatuses to reveal and
extricate data from servers and web2 reports, and it permits organizations to
get to both organized and unstructured information from browser activities,
server logs, website and link structure, page content and different sources.
```

2.1.2 b. Display the total number of terms and term frequency of each term present in them after applying stop word removal.

[83]:
```
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('english'))
word_tokens = word_tokenize(raw)
filtered_sentence = [w for w in word_tokens if not w in stop_words]
```

```
filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)
print(len(filtered_sentence))
```

```
54
```

[84]:
```
d = {}

for i in filtered_sentence:
    if i in d:
        d[i] += 1
    else:
        d[i] = 1
for key in list(d.keys()):
     print(key, ":", d[key])
```

```
Web : 2
mining : 3
application : 1
data : 2
techniques : 1
discover : 1
patterns : 1
World : 1
Wide : 1
. : 3
As : 1
name : 1
proposes : 1
, : 5
information : 2
gathered : 1
web : 1
It : 1
makes : 1
utilization : 1
automated : 1
apparatuses : 1
reveal : 1
extricate : 1
servers : 1
web2 : 1
reports : 1
permits : 1
```

```
organizations : 1
get : 1
organized : 1
unstructured : 1
browser : 1
activities : 1
server : 1
logs : 1
website : 1
link : 1
structure : 1
page : 1
content : 1
different : 1
sources : 1
```

2.1.3 c. Remove all the special characters/symbols present in the content by adding those characters as stopwords in the existing stopword list..

[85]:
```
STOP_WORDS = set(stopwords.words('english'))
STOP_WORDS.add('-')
STOP_WORDS.add(',')
STOP_WORDS.add("\'s")
STOP_WORDS.add("\'")
STOP_WORDS.add('.')
STOP_WORDS.add(',')
STOP_WORDS.add('(')
STOP_WORDS.add(')')
STOP_WORDS.add('[')
STOP_WORDS.add(']')
STOP_WORDS.add(';')
STOP_WORDS.add(':')
STOP_WORDS.add('^')
STOP_WORDS.add('..')
STOP_WORDS.add('&')
STOP_WORDS.add('=')
STOP_WORDS.add('?')

word_tokens = word_tokenize(raw)
filtered_sentence1 = [w for w in word_tokens if not w in STOP_WORDS]
print(len(filtered_sentence1))
```

```
46
```

2.1.4 d. Also, apply stemming (don't use porter stemmer) and lemmatization to the same document and display the number of terms along with their corresponding stemmed as well as lemmatized words present in them using Pandas dataframe as per the format given below

[86]:
```
from nltk.stem.snowball import SnowballStemmer
from nltk.stem import WordNetLemmatizer
ps = SnowballStemmer("english")
lemmatizer = WordNetLemmatizer()
stems = []
lemma = []
for w in filtered_sentence1:
    print(w,"-",ps.stem(w), " - ", lemmatizer.lemmatize(w))
    stems.append(ps.stem(w))
    lemma.append(lemmatizer.lemmatize(w))
```

```
Web - web  -  Web
mining - mine  -  mining
application - applic  -  application
data - data  -  data
mining - mine  -  mining
techniques - techniqu  -  technique
discover - discov  -  discover
patterns - pattern  -  pattern
World - world  -  World
Wide - wide  -  Wide
Web - web  -  Web
As - as  -  As
name - name  -  name
proposes - propos  -  proposes
information - inform  -  information
gathered - gather  -  gathered
mining - mine  -  mining
web - web  -  web
It - it  -  It
makes - make  -  make
utilization - util  -  utilization
automated - autom  -  automated
apparatuses - apparatus  -  apparatus
reveal - reveal  -  reveal
extricate - extric  -  extricate
data - data  -  data
servers - server  -  server
web2 - web2  -  web2
reports - report  -  report
permits - permit  -  permit
organizations - organ  -  organization
get - get  -  get
```

```
organized - organ  -  organized
unstructured - unstructur  -  unstructured
information - inform  -  information
browser - browser  -  browser
activities - activ  -  activity
server - server  -  server
logs - log  -  log
website - websit  -  website
link - link  -  link
structure - structur  -  structure
page - page  -  page
content - content  -  content
different - differ  -  different
sources - sourc  -  source
```

[87]:
```
data={"Original Term":filtered_sentence1,"Stemmed Term":stems,"Lemmatized Term":
 ↪lemma}
import pandas
pandas.DataFrame(data)
```

[87]:

	Original Term	Stemmed Term	Lemmatized Term
0	Web	web	Web
1	mining	mine	mining
2	application	applic	application
3	data	data	data
4	mining	mine	mining
5	techniques	techniqu	technique
6	discover	discov	discover
7	patterns	pattern	pattern
8	World	world	World
9	Wide	wide	Wide
10	Web	web	Web
11	As	as	As
12	name	name	name
13	proposes	propos	proposes
14	information	inform	information
15	gathered	gather	gathered
16	mining	mine	mining
17	web	web	web
18	It	it	It
19	makes	make	make
20	utilization	util	utilization
21	automated	autom	automated
22	apparatuses	apparatus	apparatus
23	reveal	reveal	reveal
24	extricate	extric	extricate
25	data	data	data
26	servers	server	server

27	web2	web2	web2
28	reports	report	report
29	permits	permit	permit
30	organizations	organ	organization
31	get	get	get
32	organized	organ	organized
33	unstructured	unstructur	unstructured
34	information	inform	information
35	browser	browser	browser
36	activities	activ	activity
37	server	server	server
38	logs	log	log
39	website	websit	website
40	link	link	link
41	structure	structur	structure
42	page	page	page
43	content	content	content
44	different	differ	different
45	sources	sourc	source

2.1.5 e. Count the total number of stemmed and lemmatized words.

[88]:
```
frequency1 = {}
for word in stems:
    count = frequency1.get(word,0)
    frequency1[word] = count + 1
frequency_list1 = frequency1.keys()
print("stemmed words-",len(frequency_list1))
```

```
stemmed words- 38
```

[89]:
```
frequency2 = {}
for word in lemma:
    count = frequency2.get(word,0)
    frequency2[word] = count + 1
frequency_list2 = frequency2.keys()
print("lemmatized words-",len(frequency_list2))
```

```
lemmatized words- 40
```

2.1.6 f. Display the POS tag (sentence-wise) for all the stopwords (excluding the special character/symbols), which are removed from the content, using pandas dataframe as per the format given below:

[90]:
```
import nltk
nltk.download('averaged_perceptron_tagger')
finalwords = []

for line in filtered_sentence1:
        for word in line.split():
            finalwords.append(word)
tagged = nltk.pos_tag(finalwords)
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Sreemanth\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

[91]:
```
import pandas as pd
pd.DataFrame(tagged,columns=['List of Stopwords','POS Tags'])
#print(df)
```

[91]:

	List of Stopwords	POS Tags
0	Web	NNP
1	mining	NN
2	application	NN
3	data	NNS
4	mining	NN
5	techniques	NNS
6	discover	IN
7	patterns	NNS
8	World	NNP
9	Wide	NNP
10	Web	NNP
11	As	IN
12	name	NN
13	proposes	VBZ
14	information	NN
15	gathered	VBD
16	mining	NN
17	web	NN
18	It	PRP
19	makes	VBZ
20	utilization	NN
21	automated	JJ
22	apparatuses	NNS
23	reveal	VBP
24	extricate	JJ

25	data	NNS
26	servers	NNS
27	web2	VBP
28	reports	NNS
29	permits	NNS
30	organizations	NNS
31	get	VBP
32	organized	VBN
33	unstructured	JJ
34	information	NN
35	browser	NN
36	activities	NNS
37	server	VBP
38	logs	NNS
39	website	JJ
40	link	NN
41	structure	NN
42	page	NN
43	content	JJ
44	different	JJ
45	sources	NNS

2.2 2nd Question

2.2.1 a. Extract the contents (excluding any tags) from two websites (https://en.wikipedia.org/wiki/Web_mining&https://en.wikipedia.org/wiki/Data_mining).

[92]:
```
from urllib import request
import re
```

[93]:
```
r1=request.urlopen('https://en.wikipedia.org/wiki/Web_mining').read().
 ↪decode('utf8')
```

[94]:
```
from bs4 import BeautifulSoup
soup1 = BeautifulSoup(r1,'html.parser')
```

[95]:
```
resulttext1 = soup1.find('p')
resulttext1.get_text()
# soup1 = BeautifulSoup(resulttext,'html.parser').get_text()
```

[95]:
```
'Web mining is the application of data mining techniques to discover patterns
from the World Wide Web. As the name proposes, this is information gathered by
mining the web. It makes utilization of automated apparatuses to reveal and
extricate data from servers and web2 reports, and it permits organizations to
get to both organized and unstructured information from browser activities,
server logs, website and link structure, page content and different sources.\n'
```

[96]:
```
r2=request.urlopen('https://en.wikipedia.org/wiki/Data_mining').read().
 ↪decode('utf8')
from bs4 import BeautifulSoup
soup2 = BeautifulSoup(r2,'html.parser')
resulttext2 = soup2.find('p')
resulttext2.get_text()
```

[96]:
```
'Data mining is a process of discovering patterns in large data sets involving
methods at the intersection of machine learning, statistics, and database
systems.[1] Data mining is an interdisciplinary subfield of computer science and
statistics with an overall goal to extract information (with intelligent
methods) from a data set and transform the information into a comprehensible
structure for further use.[1][2][3][4] Data mining is the analysis step of the
"knowledge discovery in databases" process, or KDD.[5] Aside from the raw
analysis step, it also involves database and data management aspects, data pre-
processing, model and inference considerations, interestingness metrics,
complexity considerations, post-processing of discovered structures,
visualization, and online updating.[1]\n'
```

2.2.2 b. Remove stopwords (including the special characters/symbols) from the contents retrieved from those two URLs and save the contents in two separate .doc file.

[97]:
```
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import sent_tokenize, word_tokenize
import pandas as pd
import re
stop_words = (stopwords.words('english'))
```

[98]:
```
stop_words.append('.')
stop_words.append(',')
stop_words.append('/')
stop_words.append('!')
stop_words.append('@')
stop_words.append('#')
stop_words.append('$')
stop_words.append('%')
stop_words.append('^')
stop_words.append('&')
stop_words.append('*')
stop_words.append('(')
stop_words.append(')')
stop_words.append('_')
stop_words.append('-')
stop_words.append('+')
stop_words.append('=')
```

```
stop_words.append('[')
stop_words.append(']')
stop_words.append(';')
stop_words.append('{')
stop_words.append('}')
stop_words.append('~')
stop_words.append('`')
stop_words.append('``')
stop_words.append("''")
word_tokens1 = word_tokenize(resulttext1.get_text())
word_tokens2 = word_tokenize(resulttext2.get_text())
```

[99]:
```
word_tokens1,word_tokens2
```

[99]:
```
(['Web',
  'mining',
  'is',
  'the',
  'application',
  'of',
  'data',
  'mining',
  'techniques',
  'to',
  'discover',
  'patterns',
  'from',
  'the',
  'World',
  'Wide',
  'Web',
  '.',
  'As',
  'the',
  'name',
  'proposes',
  ',',
  'this',
  'is',
  'information',
  'gathered',
  'by',
  'mining',
  'the',
  'web',
  '.',
  'It',
  'makes',
```

```
 'utilization',
 'of',
 'automated',
 'apparatuses',
 'to',
 'reveal',
 'and',
 'extricate',
 'data',
 'from',
 'servers',
 'and',
 'web2',
 'reports',
 ',',
 'and',
 'it',
 'permits',
 'organizations',
 'to',
 'get',
 'to',
 'both',
 'organized',
 'and',
 'unstructured',
 'information',
 'from',
 'browser',
 'activities',
 ',',
 'server',
 'logs',
 ',',
 'website',
 'and',
 'link',
 'structure',
 ',',
 'page',
 'content',
 'and',
 'different',
 'sources',
 '.'],
['Data',
 'mining',
```

```
'is',
'a',
'process',
'of',
'discovering',
'patterns',
'in',
'large',
'data',
'sets',
'involving',
'methods',
'at',
'the',
'intersection',
'of',
'machine',
'learning',
',',
'statistics',
',',
'and',
'database',
'systems',
'.',
'[',
'1',
']',
'Data',
'mining',
'is',
'an',
'interdisciplinary',
'subfield',
'of',
'computer',
'science',
'and',
'statistics',
'with',
'an',
'overall',
'goal',
'to',
'extract',
'information',
'(',
```

```
'with',
'intelligent',
'methods',
')',
'from',
'a',
'data',
'set',
'and',
'transform',
'the',
'information',
'into',
'a',
'comprehensible',
'structure',
'for',
'further',
'use',
'.',
'[',
'1',
']',
'[',
'2',
']',
'[',
'3',
']',
'[',
'4',
']',
'Data',
'mining',
'is',
'the',
'analysis',
'step',
'of',
'the',
'``',
'knowledge',
'discovery',
'in',
'databases',
"''",
'process',
```

```
',',
'or',
'KDD',
'.',
'[',
'5',
']',
'Aside',
'from',
'the',
'raw',
'analysis',
'step',
',',
'it',
'also',
'involves',
'database',
'and',
'data',
'management',
'aspects',
',',
'data',
'pre-processing',
',',
'model',
'and',
'inference',
'considerations',
',',
'interestingness',
'metrics',
',',
'complexity',
'considerations',
',',
'post-processing',
'of',
'discovered',
'structures',
',',
'visualization',
',',
'and',
'online',
'updating',
```

```
  '.',
  '[',
  '1',
  ']'])
```

[100]:
```
words1 = [w for w in word_tokens1 if not w in stop_words]
words2 = [w for w in word_tokens2 if not w in stop_words]
```

[101]:
```
pd.DataFrame({"doc1":words1})
```

[101]:

	doc1
0	Web
1	mining
2	application
3	data
4	mining
5	techniques
6	discover
7	patterns
8	World
9	Wide
10	Web
11	As
12	name
13	proposes
14	information
15	gathered
16	mining
17	web
18	It
19	makes
20	utilization
21	automated
22	apparatuses
23	reveal
24	extricate
25	data
26	servers
27	web2
28	reports
29	permits
30	organizations
31	get
32	organized
33	unstructured
34	information
35	browser
36	activities

```
37         server
38           logs
39        website
40           link
41      structure
42           page
43        content
44      different
45        sources
```

[102]:
```
pd.DataFrame({"doc2":words2})
```

[102]:
```
             doc2
0            Data
1          mining
2         process
3     discovering
4        patterns
..             ...
72     structures
73  visualization
74         online
75       updating
76              1

[77 rows x 1 columns]
```

[103]:
```
with open('doc1.doc', 'w') as f:
    for item in words1:
        f.write("%s\n" % item)
with open('doc2.doc', 'w') as f:
    for item in words2:
        f.write("%s\n" % item)
```

2.2.3 c. Display the Term-Document incidence matrix using Boolean, Bag-of-words and Complete representation (Use pandas dataframe).

[104]:
```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
```

[105]:
```

readwords1 = []
readwords2 = []
# opening the text file
with open('doc1.doc','r') as file:

    # reading each line
    for line in file:
```

```
        # reading each word
        for word in line.split():

            # displaying the words
            readwords1.append(word)
# opening the text file
with open('doc2.doc','r') as file:

    # reading each line
    for line in file:

        # reading each word
        for word in line.split():

            # displaying the words
            readwords2.append(word)
```

[106]:
```
text1 = " ".join(readwords1)
text2 = " ".join(readwords2)
```

[107]:
```
text1
```

[107]:
```
'Web mining application data mining techniques discover patterns World Wide Web
As name proposes information gathered mining web It makes utilization automated
apparatuses reveal extricate data servers web2 reports permits organizations get
organized unstructured information browser activities server logs website link
structure page content different sources'
```

[108]:
```
text2
```

[108]:
```
'Data mining process discovering patterns large data sets involving methods
intersection machine learning statistics database systems 1 Data mining
interdisciplinary subfield computer science statistics overall goal extract
information intelligent methods data set transform information comprehensible
structure use 1 2 3 4 Data mining analysis step knowledge discovery databases
process KDD 5 Aside raw analysis step also involves database data management
aspects data pre-processing model inference considerations interestingness
metrics complexity considerations post-processing discovered structures
visualization online updating 1'
```

[109]:
```
docs = [text1,text2]
```

2.2.4 Binary Representation

[114]:
```
booldoc1 = []
booldoc2 = []
for x in (text1 + text2).split(" "):
    if(x in text1.split(" ")):
```

```
        booldoc1.append(1)
    else:
        booldoc1.append(0)
for x in (text1 + text2).split(" "):
    if(x in text2.split(" ")):
        booldoc2.append(1)
    else:
        booldoc2.append(0)
booldata = {"Words":(text1 + text2).split(" "),"Doc 1":booldoc1,"Doc 2":
 ↪booldoc2}
pd.DataFrame(booldata)
```

[114]:

	Words	Doc 1	Doc 2
0	Web	1	0
1	mining	1	1
2	application	1	0
3	data	1	1
4	mining	1	1
..	...	...	...
117	structures	0	1
118	visualization	0	1
119	online	0	1
120	updating	0	1
121	1	0	1

[122 rows x 3 columns]

[136]:
```
vec = CountVectorizer()
```

[137]:
```
X = vec.fit_transform(docs)
X = X.transpose()
```

[138]:
```
#df =pd.DataFrame(X.toarray(), columns=vec.get_feature_names())
#pd.DataFrame(X.toarray(), columns=vec.get_feature_names())
df =pd.DataFrame(X.toarray(), columns=['doc1 Freq','doc2 Freq'])
df['words'] = vec.get_feature_names()
```

2.2.5 BOW Representation

[118]:
```
#print(df)
df[['words','doc1 Freq','doc2 Freq']]
```

[118]:

	words	doc1 Freq	doc2 Freq
0	activities	1	0
1	also	0	1
2	analysis	0	2
3	apparatuses	1	0
4	application	1	0
..	...	...	...

```
85          web          3          0
86         web2          1          0
87      website          1          0
88         wide          1          0
89        world          1          0

[90 rows x 3 columns]
```

2.2.6 Complete Representation

[119]:
```
compwords1 =[]
pos1=[]
for x in resulttext1.get_text().split(" "):
    m =0
    if x in words1:
        for y in text1.split(" "):
            m += 1
            if(x==y):
                compwords1.append(x)
                pos1.append(text1.index(x,m-1))
finaldata1 = {"words":compwords1,"Position":pos1}
h =pd.DataFrame(finaldata1)
h.drop_duplicates(subset=['words', 'Position'],inplace=True)
h.head()
```

[119]:
```
         words  Position
0          Web         0
1          Web        75
2       mining         4
4       mining        28
5  application        11
```

[120]:
```
compwords2 =[]
pos2=[]
for x in resulttext1.get_text().split(" "):
    m =0
    if x in words2:
        for y in text2.split(" "):
            m += 1
            if(x==y):
                compwords2.append(x)
                pos2.append(text2.index(x,m-1))
finaldata2 = {"words":compwords2,"Position":pos2}
h = pd.DataFrame(finaldata2)
h.drop_duplicates(subset=['words', 'Position'],inplace=True)
h.head()
```

[120]:

	words	Position
0	mining	5
1	mining	140
3	data	47
5	data	116
10	patterns	32

2.2.7 d. Input a search a query (preferably a sentence) and compare the contents of the both pages with the processed query. Display the similarity result based on highest frequency matching count of the term.

[121]:
```
searchstring = "Web hello intersection data is analysis interdisciplinary␣
 ↪subfield usefull"

sc1=0
sc2=0
for x in searchstring.split(" "):
        if(x in text1.split(" ")):
            sc1=sc1+1
        if(x in text2.split(" ")):
            sc2=sc2+1

if(sc1>sc2):
    print("Doc1 is more relevent")
elif(sc1==sc2):
    print("Both Documents are equally relevent")
elif(sc1<sc2):
    print("Doc2 is more relevent")
```

```
Doc2 is more relevent
```

2.3 Question 3

2.3.1 Write a python program to prepare the Word Clouds representation based on the content present in the two document files prepared in Q.No. 2. A sample Word Clouds representation is provided below for reference

[122]:
```
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS
```

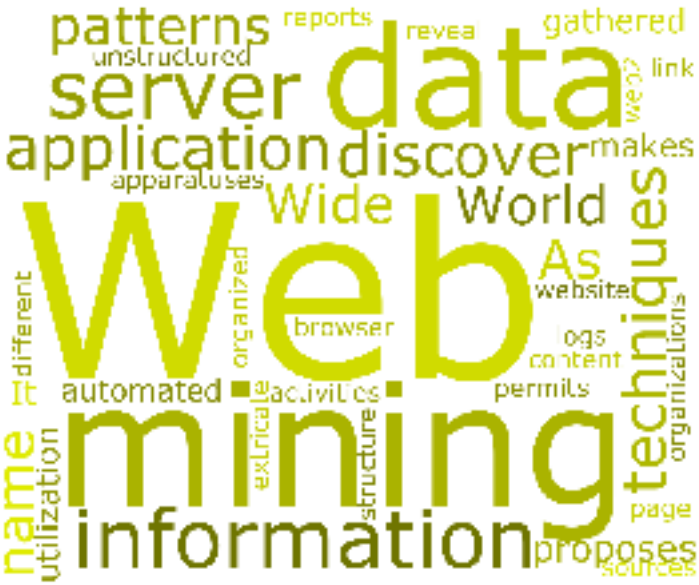
[123]:
```
def random_color_func(word=None, font_size=None, position=None,␣
 ↪orientation=None, font_path=None, random_state=None):
    h = int(360.0 * 45.0 / 255.0)
    s = int(100.0 * 255.0 / 255.0)
    l = int(100.0 * float(random_state.randint(60, 120)) / 255.0)

    return "hsl({}, {}%, {}%)".format(h, s, l)
```

[124]:
```
file_content1=open ("doc1.doc").read()
file_content2=open ("doc1.doc").read()
```

[125]:
```
wordcloud1 = WordCloud(font_path = r'C:\Windows\Fonts\Verdana.ttf',
                       stopwords = STOPWORDS,
                       background_color = 'white',
                       width = 1200,
                       height = 1000,
                       color_func = random_color_func
                       ).generate(file_content1)
wordcloud2 = WordCloud(font_path = r'C:\Windows\Fonts\Verdana.ttf',
                       stopwords = STOPWORDS,
                       background_color = 'white',
                       width = 1200,
                       height = 1000,
                       color_func = random_color_func
                       ).generate(file_content2)
```

[126]:
```
plt.imshow(wordcloud1)
plt.axis('off')
plt.show()
```



[127]:
```
plt.imshow(wordcloud2)
plt.axis('off')
plt.show()
```

2.4 Question 4

2.4.1 Write a python program to show the implementation of sentence paraphrasing through synonyms (retaining semantic meaning) for the following four sentences. Display at least three other paraphrased sentences for each sentence mentioned below.

a. The quick brown fox jumps over the lazy dog

b. Obama and Putin met the previous week

c. At least 12 people were killed in the battle last week

d. I will go home and come back tomorrow.

[128]:
```
from nltk.tokenize import word_tokenize
from nltk.tag import pos_tag
from nltk.corpus import wordnet as wn
import pandas as pd
def tag(sentence):
 words = word_tokenize(sentence)
 words = pos_tag(words)
 return words

def paraphraseable(tag):
 return tag.startswith('NN') or tag == 'VB' or tag.startswith('JJ')

def pos(tag):
```

```
 if tag.startswith('NN'):
  return wn.NOUN
 elif tag.startswith('V'):
  return wn.VERB

def synonyms(word, tag):
    lemma_lists = [ss.lemmas() for ss in wn.synsets(word, pos(tag))]
    lemmas = [lemma.name() for lemma in sum(lemma_lists, [])]
    return set(lemmas)

def synonymIfExists(sentence):
 for (word, t) in tag(sentence):
   if paraphraseable(t):
    syns = synonyms(word, t)
    if syns:
     if len(syns) > 1:
      yield [word, list(syns)]
      continue
   yield [word, []]

def paraphrase(sentence):
 return [x for x in synonymIfExists(sentence)]
```

[129]:
```
s1 = paraphrase("The quick brown fox jumps over the lazy dog")
s2 = paraphrase("Obama and Putin met the previous week")
s3 = paraphrase("At least 12 people were killed in the battle last week")
s4 = paraphrase("I will go home and come back tomorrow")
```

[130]:
```
s1,s2,s3,s4
```

[130]:
```
([['The', []],
  ['quick',
   ['spry',
    'promptly',
    'ready',
    'prompt',
    'straightaway',
    'fast',
    'immediate',
    'flying',
    'nimble',
    'warm',
    'quickly',
    'speedy',
    'quick',
    'agile']],
  ['brown',
   ['Robert_Brown',
```

```
   'Brown',
   'brownness',
   'brown',
   'Brown_University',
   'John_Brown']],
 ['fox',
  ['George_Fox', 'slyboots', 'dodger', 'Charles_James_Fox', 'Fox', 'fox']],
 ['jumps', []],
 ['over', []],
 ['the', []],
 ['lazy', ['faineant', 'indolent', 'otiose', 'work-shy', 'lazy', 'slothful']],
 ['dog',
  ['weenie',
   'heel',
   'cad',
   'andiron',
   'frank',
   'wienerwurst',
   'bounder',
   'pawl',
   'frump',
   'hound',
   'dog-iron',
   'blackguard',
   'firedog',
   'wiener',
   'Canis_familiaris',
   'hotdog',
   'detent',
   'frankfurter',
   'dog',
   'domestic_dog',
   'click',
   'hot_dog']]],
[['Obama', []],
 ['and', []],
 ['Putin', ['Vladimir_Vladimirovich_Putin', 'Vladimir_Putin', 'Putin']],
 ['met', []],
 ['the', []],
 ['previous', ['old', 'late', 'former', 'previous', 'premature']],
 ['week', ['calendar_week', 'week', 'workweek', 'hebdomad']]],
[['At', []],
 ['least', ['least', 'to_the_lowest_degree']],
 ['12', []],
 ['people',
  ['citizenry',
   'multitude',
```

```
   'masses',
   'hoi_polloi',
   'mass',
   'people',
   'the_great_unwashed']],
 ['were', []],
 ['killed', []],
 ['in', []],
 ['the', []],
 ['battle', ['engagement', 'conflict', 'battle', 'fight', 'struggle']],
 ['last',
  ['live',
   'finally',
   'concluding',
   'last-place',
   'hold_up',
   'lastly',
   'finale',
   'conclusion',
   'final',
   'end',
   'net',
   'lowest',
   'death',
   'utmost',
   "cobbler's_last",
   'finish',
   'terminal',
   'finis',
   'stopping_point',
   'survive',
   'close',
   "shoemaker's_last",
   'final_stage',
   'live_on',
   'hold_out',
   'go',
   'in_conclusion',
   'last',
   'endure']],
 ['week', ['calendar_week', 'week', 'workweek', 'hebdomad']]],
[['I', []],
 ['will', []],
 ['go',
  ['perish',
   'fail',
   'go_bad',
```

```
'drop_dead',
'start',
'endure',
'live',
'die',
'buy_the_farm',
'extend',
'work',
'hold_up',
'rifle',
'give_out',
'give-up_the_ghost',
'exit',
'belong',
'conk',
'decease',
'blend',
'give_way',
'lead',
'proceed',
'run',
'pass_away',
'break',
'get_going',
'locomote',
'blend_in',
'become',
'pop_off',
'snuff_it',
'plump',
'travel',
'pass',
'depart',
'break_down',
'run_low',
'survive',
'conk_out',
'fit',
'function',
'go_away',
'choke',
'live_on',
'hold_out',
'go',
'get',
'move',
'sound',
```

```
  'kick_the_bucket',
  "cash_in_one's_chips",
  'last',
  'run_short',
  'expire',
  'operate',
  'croak']],
['home',
 ['rest_home',
  'abode',
  'house',
  'plate',
  'base',
  'home',
  'home_plate',
  'nursing_home',
  'habitation',
  'menage',
  'domicile',
  'dwelling',
  'place',
  'family',
  'home_base',
  'dwelling_house',
  'household']],
['and', []],
['come',
 ['come_up',
  'amount',
  'add_up',
  'fare',
  'do',
  'issue_forth',
  'fall',
  'descend',
  'derive',
  'make_out',
  'come_in',
  'occur',
  'hail',
  'come',
  'arrive',
  'get_along',
  'follow',
  'get',
  'number',
  'total']],
```

```
 ['back', []],
 ['tomorrow', []]])
```

[131]:
```
def parphr(arr,j):
    s1 = ""

    for i in range(len(arr)):
        if(arr[i][1]==[]):
            wrd=arr[i][0]
        elif(len(arr[i][1])<=j):
            wrd=arr[i][0]
        else:
            wrd=arr[i][1][j]

        s1=s1+" "+wrd
    return s1
```

[132]:
```
for i in range(3):
    print(parphr(s1,i))
```

```
The spry Robert_Brown George_Fox jumps over the faineant weenie
The promptly Brown slyboots jumps over the indolent heel
The ready brownness dodger jumps over the otiose cad
```

[133]:
```
for i in range(3):
    print(parphr(s2,i))
```

```
Obama and Vladimir_Vladimirovich_Putin met the old calendar_week
Obama and Vladimir_Putin met the late week
Obama and Putin met the former workweek
```

[134]:
```
for i in range(3):
    print(parphr(s3,i))
```

```
At least 12 citizenry were killed in the engagement live calendar_week
At to_the_lowest_degree 12 multitude were killed in the conflict finally week
At least 12 masses were killed in the battle concluding workweek
```

[135]:
```
for i in range(3):
    print(parphr(s4,i))
```

```
I will perish rest_home and come_up back tomorrow
I will fail abode and amount back tomorrow
I will go_bad house and add_up back tomorrow
```

[ ]:

[ ]:

[ ]: