CS 4395 Human Language Technologies Assignment 7 Wordnet Professor Mazidi 9/21/2022

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('sentiwordnet')
from nltk.corpus import wordnet as wn
from nltk.corpus import sentiwordnet as swn
nltk.download('book')
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
```

```
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/universal_tagset.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
```

```
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Unzipping help/tagsets.zip.
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection book
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
```

Question 1

WordNet is a lexical database of English words that exists for use in natural language processing. It consists of records that hold data regarding the semantic relations between words, such as synonyms, antonyms, etc. Its useful for quickly determining the relations between words in order to extract information from them with relatively little work on your part.

Question 2

```
wn.synsets('book')
```

```
[Synset('book.n.01'),
 Synset('book.n.02'),
 Synset('record.n.05'),
 Synset('script.n.01'),
 Synset('ledger.n.01'),
 Synset('book.n.06'),
 Synset('book.n.07'),
 Synset('koran.n.01'),
 Synset('bible.n.01'),
 Synset('book.n.10'),
 Synset('book.n.11'),
 Synset('book.v.01'),
 Synset('reserve.v.04'),
 Synset('book.v.03'),
 Synset('book.v.04')]
```

Question 3

Verbs in the wordnet (words in general) seem to be grouped with words that have similar definitions. The verb form of 'book' is grouped with 'reserve' and other verb forms of book, for

```
wn.synset('book.v.03').definition()

     'record a charge in a police register'


wn.synset('book.v.03').examples()

     ['The policeman booked her when she tried to solicit a man']


wn.synset('book.v.03').lemmas()

     [Lemma('book.v.03.book')]


# iterate over synsets
book_synsets = wn.synsets('book', pos=wn.VERB)
for sense in book_synsets:
    lemmas = [l.name() for l in sense.lemmas()]
    print("Synset: " + sense.name() + "(" +sense.definition() + ")  \n\t Lemmas:" + str(lemma

     Synset: book.v.01(engage for a performance)
              Lemmas:['book']
     Synset: reserve.v.04(arrange for and reserve (something for someone else) in advance)
              Lemmas:['reserve', 'hold', 'book']
     Synset: book.v.03(record a charge in a police register)
              Lemmas:['book']
     Synset: book.v.04(register in a hotel booker)
              Lemmas:['book']
```

Question 4

'book' has no meronyms, holonyms, and antonyms.

```
print(wn.synset('book.v.03').hypernyms())
wn.synset('book.v.03').hyponyms()
# wn.synset('book.n.01').meronyms()
# wn.synset('book.n.01').holonyms()
# wn.synset('book.n.01').antonyms()

     [Synset('record.v.01')]
     [Synset('ticket.v.01')]
```

Question 5

```
wn.synsets('walk')
```

```
[Synset('walk.n.01'),
 Synset('base_on_balls.n.01'),
 Synset('walk.n.03'),
 Synset('walk.n.04'),
 Synset('walk.n.05'),
 Synset('walk.n.06'),
 Synset('walk_of_life.n.01'),
 Synset('walk.v.01'),
 Synset('walk.v.02'),
 Synset('walk.v.03'),
 Synset('walk.v.04'),
 Synset('walk.v.05'),
 Synset('walk.v.06'),
 Synset('walk.v.07'),
 Synset('walk.v.08'),
 Synset('walk.v.09'),
 Synset('walk.v.10')]
```

Question 6

```
wn.synset('walk.n.03').definition()
```

```
'manner of walking'
```

```
wn.synset('walk.n.03').examples()
```

```
['he had a funny walk']
```

```
wn.synset('walk.n.03').lemmas()
```

```
[Lemma('walk.n.03.walk'), Lemma('walk.n.03.manner_of_walking')]
```

```
# iterate over synsets
walk_synsets = wn.synsets('walk', pos=wn.VERB)
for sense in walk_synsets:
    lemmas = [l.name() for l in sense.lemmas()]
    print("Synset: " + sense.name() + "(" +sense.definition() + ")  \n\t Lemmas:" + str(lemma
```

```
Synset: walk.v.01(use one's feet to advance; advance by steps)
         Lemmas:['walk']
Synset: walk.v.02(accompany or escort)
```

```
         Lemmas:['walk']
Synset: walk.v.03(obtain a base on balls)
         Lemmas:['walk']
Synset: walk.v.04(traverse or cover by walking)
         Lemmas:['walk']
Synset: walk.v.05(give a base on balls to)
         Lemmas:['walk']
Synset: walk.v.06(live or behave in a specified manner)
         Lemmas:['walk']
Synset: walk.v.07(be or act in association with)
         Lemmas:['walk']
Synset: walk.v.08(walk at a pace)
         Lemmas:['walk']
Synset: walk.v.09(make walk)
         Lemmas:['walk']
Synset: walk.v.10(take a walk; go for a walk; walk for pleasure)
         Lemmas:['walk', 'take_the_air']
```

Question 7

```
wn.morphy('denies')
```

```
'deny'
```

Question 8

The implementations of the Wu-Palmer and Lesk algorithm seem to be rather accurate. The Wu-Palmer algoritm returns a value for how similar two usages of a word are, which can be very useful for extracting information from a sentence. The lesk algorithm is much more interesting. It accurately returns the exact synset which is extremely useful for quickly gaining the context of a words usage in the passage being processed. I will for sure use this more often in the future.

```
# I will use words 'achieve' and 'attain'
print(wn.synsets('accomplish'), '\n')
wn.synsets('attain')
```

```
[Synset('carry_through.v.01'), Synset('achieve.v.01')]

[Synset('achieve.v.01'),
 Synset('reach.v.02'),
 Synset('fall_upon.v.01'),
 Synset('reach.v.01')]
```

```
# Wu-Palmer algorithm
achieve = wn.synset('achieve.v.01')
```

```
reach = wn.synset('reach.v.02')
wn.wup_similarity(achieve, reach)

     0.3333333333333333

# lesk algorithm
from nltk.wsd import lesk
sent = ['I', 'will', 'achieve', 'my', 'goals']
print(lesk(sent, 'achieve'))

     Synset('achieve.v.01')


# printing out definition and example of the returned synset to check for accuracy
print(wn.synset('achieve.v.01').definition(), '\n')
wn.synset('achieve.v.01').examples()

     to gain with effort

     ['she achieved her goal despite setbacks']
```

Question 9

The sentiwordnet is an extension of wordnet that stores data about each synset's sentiments, or how positiive or negative each synset is. Each synset is assigned three scores: positivity, negativity, and objectivity. The scores are between 0-1 and all three scores always add up to 1.

```
# testing sentiwordnet
senti_list = list(swn.senti_synsets('frantic'))
for item in senti_list:
    print(item)

     <frantic.s.01: PosScore=0.25 NegScore=0.5>
     <delirious.s.02: PosScore=0.5 NegScore=0.25>


# outputting polarity for each word in a sentence
sent = 'We love chinese food'
neg = 0
pos = 0
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        print(syn, ' = ', syn.neg_score())
        print(syn, ' = ', syn.pos_score())

     <love.n.01: PosScore=0.625 NegScore=0.0>  =  0.0
```

```
<love.n.01: PosScore=0.625 NegScore=0.0>  =  0.625
<chinese.n.01: PosScore=0.0 NegScore=0.0>  =  0.0
<chinese.n.01: PosScore=0.0 NegScore=0.0>  =  0.0
<food.n.01: PosScore=0.0 NegScore=0.0>  =  0.0
<food.n.01: PosScore=0.0 NegScore=0.0>  =  0.0
```

The scores seem to be somewhat arbitrary on my end, but there is likely a proper algorithm for this under the hood. Understanding how positive or negative a passage is can tell you a lot about the intention behind the passage in addition to some other information.

Question 10

A collocation occurs when multiple words are combined to convey a different meaning. For example, 'bright idea' has a different meaning than each of the two words individually.

```
from nltk.book import *
print(text4)
print(text4.collocations())
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
<Text: Inaugural Address Corpus>
United States; fellow citizens; years ago; four years; Federal
Government; General Government; American people; Vice President; God
bless; Chief Justice; one another; fellow Americans; Old World;
Almighty God; Fellow citizens; Chief Magistrate; every citizen; Indian
tribes; public debt; foreign nations
None
```

```
text = ' '.join(text4.tokens)
text[:50]
```

```
'Fellow - Citizens of the Senate and of the House o'
```

```
import math
vocab = len(set(text4))
```

```
hg = text.count('Federal Government')/vocab
print("p(Federal Government) = ",hg )
h = text.count('Federal')/vocab
print("p(Federal) = ", h)
g = text.count('Government')/vocab
print('p(Government) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
p(Federal Government) =  0.0031920199501246885
p(Federal) =  0.006483790523690773
p(Government) =  0.033715710723192O2
pmi =  3.868067366919006
```

It seems the term 'federal government' contains relatively little information based on its occurance, but that's not too surprising. Perhaps its inaccurate because the term "government" could be used to mean federal government in certain contexts, but this won't be picked up by mutual information