Afaf Guesmia

ID: axg190061

Assignmnet3

1.WordNet is defined as a lexical database of verbs, nouns, adjectives, and adverbs. The goal of the project is to support some theories of human semantic memory, which talk about how humans organize concepts mentally in a type of hierarchy. Verbs, nouns, adjectives are grouped into sets of cognitive synonyms.In this example we used NLTK as it provides a WordNet interface.

In [7]:
```
from nltk.corpus import wordnet as wn
wn.synsets('act')
```

Out[7]:
```
[Synset('act.n.01'),
 Synset('act.n.02'),
 Synset('act.n.03'),
 Synset('act.n.04'),
 Synset('act.n.05'),
 Synset('act.v.01'),
 Synset('act.v.02'),
 Synset('act.v.03'),
 Synset('act.v.04'),
 Synset('act.v.05'),
 Synset('act.v.06'),
 Synset('work.v.03'),
 Synset('act.v.08'),
 Synset('dissemble.v.03'),
 Synset('act.v.10')]
```

In [35]:
```
wn.synset('act.n.01').definition()
```

Out[35]:
```
'a legal document codifying the result of deliberations of a committee or society or legislative body'
```

In [47]:
```
wn.synset('act.n.01').examples()
```

Out[47]:
```
[]
```

In [38]:
```
wn.synset('act.n.01').lemmas()
```

Out[38]:
```
[Lemma('act.n.01.act'), Lemma('act.n.01.enactment')]
```

In [44]:
```
noun=wn.synset('act.n.01')
hyp = noun.hypernyms()[0]
top = wn.synset('entity.n.01')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
Synset('legal_document.n.01')
Synset('document.n.01')
Synset('writing.n.02')
Synset('written_communication.n.01')
Synset('communication.n.02')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

My observation on how the WordNet organizes the nouns, I believe is in hierarchies based on their hypernyms, and also by the similarities of their meanings. Also, I have noticed if there is a portmanteau which is a word made of two other words such as legal_document, it will still consider a document as a different noun, which I found very interesting that it can differentiate.

In [6]:
```
from nltk.corpus import wordnet as wn
noun=wn.synset('act.n.01')
print("Hypernyms:",noun.hypernyms())
print("Hyponyms:",noun.hyponyms())
print("Meronyms:",noun.part_meronyms())
print("Anatonym:", noun.lemmas()[0].antonyms())
```

```
Hypernyms: []
Hyponyms: []
Meronyms: [Synset('chromosphere.n.01'), Synset('photosphere.n.01')]
Anatonym: []
```

The WordNet has organized the verbs in a hierarchies way, starting with the verb think then remember then again think which I have found very interesting that it did not print the similar verbs together first. I have also noticed that there is only one different verb is remember.

In [74]:
```
from nltk.corpus import wordnet as wn
wn.synsets('think')
```

Out[74]:
```
[Synset('think.n.01'),
 Synset('think.v.01'),
 Synset('think.v.02'),
 Synset('think.v.03'),
 Synset('remember.v.01'),
 Synset('think.v.05'),
 Synset('think.v.06'),
 Synset('intend.v.01'),
 Synset('think.v.08'),
 Synset('think.v.09'),
 Synset('think.v.10'),
 Synset('think.v.11'),
 Synset('think.v.12'),
 Synset('think.v.13')]
```

In [76]:
```
wn.synset('think.v.01').definition()
```

Out[76]:
```
'judge or regard; look upon; judge'
```

In [77]:
```
wn.synset('think.v.01').examples()
```

Out[77]:
```
['I think he is very smart',
 "I believe her to be very smart',
 'I think that he is her boyfriend",
 'The racist conceives such people to be inferior']
```

In [78]:
```
wn.synset('think.v.01').lemmas()
```

Out[78]:
```
[Lemma('think.v.01.think'),
 Lemma('think.v.01.believe'),
 Lemma('think.v.01.consider'),
 Lemma('think.v.01.conceive')]
```

In [11]:
```
from nltk.corpus import wordnet as wn
verb=wn.synset('think.v.01')
hyp = verb.hypernyms()[0]
top = wn.synset('root.v.01')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
    else:
        break
```

```
Synset('evaluate.v.02')
Synset('think.v.03')
```

In [80]:
```
wn.morphy('think',wn.VERB)
```

Out[80]:
```
'think'
```

In [4]:
```
from nltk.corpus import wordnet as wn

wn.synsets('eat')
```

Out[4]:
```
[Synset('eat.v.01'),
 Synset('eat.v.02'),
 Synset('feed.v.06'),
 Synset('eat.v.04'),
 Synset('consume.v.05'),
 Synset('corrode.v.01')]
```

In [5]:
```
from nltk.corpus import wordnet as wn
wn.synsets('sleep')
```

Out[5]:
```
[Synset('sleep.n.01'),
 Synset('sleep.n.02'),
 Synset('sleep.n.03'),
 Synset('rest.n.05'),
 Synset('sleep.v.01'),
 Synset('sleep.v.02')]
```

The two words sleep and eat which are two verbs, have 0.25 similarity. I thought that the similar range would be higher as the similarity range from 0 to 1, I was expeting that it will be higher for the two words.

In [6]:
```
eat=wn.synset('eat.v.01')
sleep=wn.synset('sleep.v.01')
wn.wup_similarity(eat, sleep)
```

Out[6]:
```
0.25
```

SentiWordNet is defined as a lexical resource that is built on top of WordNet, which has three synsets, positivity, negativity, and objectivity. It shows how negative or positive the word is. It can be imported from NLTK as well.

In [8]:
```
import nltk
nltk.download('sentiwordnet')
```

```
[nltk_data] Downloading package sentiwordnet to
[nltk_data]     C:\Users\manny3\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\sentiwordnet.zip.
```

Out[8]:
```
True
```

In [9]:
```
from nltk.corpus import wordnet as wn
wn.synsets('anxiety')
```

Out[9]:
```
[Synset('anxiety.n.01'), Synset('anxiety.n.02')]
```

In [26]:
```
from nltk.corpus import sentiwordnet as swn

anxiety = list(swn.senti_synsets('anxiety'))
for x in anxiety:
    print("word:",x.synset.name())
    print("Positive score = ", x.pos_score())
    print("Negative score = ", x.neg_score())
    print("Objective score = ", x.obj_score())
```

```
word: anxiety.n.01
Positive score =  0.0
Negative score =  0.125
Objective score =  0.875
word: anxiety.n.02
Positive score =  0.125
Negative score =  0.75
Objective score =  0.125
```

In [28]:
```
from nltk.corpus import sentiwordnet as swn
sent='The weather is nice today'
neg = 0
pos = 0
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        for x in syn_list:
            print("word:",x.synset.name())
    print("Positive score = ", x.pos_score())
    print("Negative score = ", x.neg_score())
    print("Objective score = ", x.obj_score())
```

```
Positive score =  0.125
Negative score =  0.75
Objective score =  0.125
word: weather.n.01
word: weather.v.01
word: weather.v.02
word: weather.v.03
word: weather.v.04
word: upwind.s.01
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
word: be.v.01
word: be.v.02
word: be.v.03
word: exist.v.01
word: be.v.05
word: equal.v.01
word: constitute.v.01
word: be.v.08
word: embody.v.02
word: be.v.10
word: be.v.11
word: be.v.12
word: cost.v.01
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
word: nice.n.01
word: nice.a.01
word: decent.s.01
word: nice.s.03
word: dainty.s.04
word: courteous.s.01
Positive score =  0.75
Negative score =  0.0
Objective score =  0.25
word: today.n.01
word: today.n.02
word: nowadays.r.01
word: today.r.02
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
```

My observation for the score of the polarity of each word in the sentence, is that couple of words have a 0 positive and negative score such as the word be or the word today which is something I was not expecting, I believed it will be more positive than negative. The word nice has a 0.75 positive score and 0.0 negative score which should be accurate. I am also surprised that the word weather got a 0 for both positive and negative scores, I also was expecting it to be more on the positive side.

In [3]:
```
import nltk
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to
[nltk_data]     C:\Users\manny3\AppData\Roaming\nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
```

Out[3]:
```
True
```

Collocation is when two words ore more words usually occur together an example can be social media, or machine learning. As we know if the aspect of NLP the processing of the word is very important, and in collocation the context is important.

In [6]:
```
import nltk
from nltk.book import *
text4
```

Out[6]:
```
<Text: Inaugural Address Corpus>
```

In [7]:
```
text4.collocations()
```

```
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
```

In [10]:
```
text = ' '.join(text4.tokens)
text[:50]
```

Out[10]:
```
'Fellow - Citizens of the Senate and of the House o'
```

My observation is that the pmi of the words chief and justice do prove that there is a collocation between them. As it was discussed during the lecture if the pmi value is 0 means that the two words are independent and there is no collocation. however, in this example, the pmi is positive which proves that the two words occur together more than expected by chance.

In [11]:
```
import math
vocab = len(set(text4))
hg = text.count('Chief Justice')/vocab
print('p(Chief Justice) = ',hg )
h = text.count('Chief')/vocab
print('p(Chief) = ', h)
g = text.count('Justice')/vocab
print('p(Justice) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
p(Chief Justice) =  0.0013965508728179551
p(Chief) =  0.0027930174563591022
p(Justice) =  0.0023940149625935516
pmi =  7.706352115508489
```

In [ ]: