WordNet Assignment

Summary of WordNet:

WordNet is a database of nouns, verbs, adjectives, and adverbs that uses hierarchical organization. It lists the glosses, short definitions, the synsets, the synonym sets, use examples, and relations to other words. A synset is a synonym set of related words that represents each meaning of a word. WordNet can be used to analyze word relations like hypernyms, hyponyms, meronyms, holonyms, and troponyms.

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

wn.synsets('home')
```

Out[ ]:
```
[Synset('home.n.01'),
 Synset('dwelling.n.01'),
 Synset('home.n.03'),
 Synset('home_plate.n.01'),
 Synset('base.n.14'),
 Synset('home.n.06'),
 Synset('home.n.07'),
 Synset('family.n.01'),
 Synset('home.n.09'),
 Synset('home.v.01'),
 Synset('home.v.02'),
 Synset('home.a.01'),
 Synset('home.a.02'),
 Synset('home.s.03'),
 Synset('home.r.01'),
 Synset('home.r.02'),
 Synset('home.r.03')]
```

In [ ]:
```
print("Definition: ",wn.synset('dwelling.n.01').definition())
print("Example of usage: ",wn.synset('dwelling.n.01').examples())
print("Lemmas: ",wn.synset('dwelling.n.01').lemmas())

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

```
Definition:  housing that someone is living in
Example of usage:  ['he built a modest dwelling near the pond', 'they raise money to pro
vide homes for the homeless']
Lemmas:  [Lemma('dwelling.n.01.dwelling'), Lemma('dwelling.n.01.home'), Lemma('dwelling.
n.01.domicile'), Lemma('dwelling.n.01.abode'), Lemma('dwelling.n.01.habitation'), Lemma
('dwelling.n.01.dwelling_house')]
Synset('housing.n.01')
Synset('structure.n.01')
Synset('artifact.n.01')
Synset('whole.n.02')
```

```
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Noun organization in WordNet

WordNet synsets are connect by semantic relations that are hierarchical and nouns are the most highly connected. There is a uniform top level for nouns in WordNet which is "entity" where, for every noun that is traced back up the levels, it will get to 'entity' eventually. This can be seen with the noun dwelling used above

In [ ]:
```
print("Hypernyms: ",wn.synset('dwelling.n.01').hypernyms())
print("Hyponyms: ", wn.synset('dwelling.n.01').hyponyms())
print("Meronyms: ",wn.synset('dwelling.n.01').part_meronyms())
print("Holonynyms: ",wn.synset('dwelling.n.01').part_holonyms())
dwelling = wn.synsets('dwelling',pos=wn.NOUN)[0]
print("Antonyms: ",dwelling.lemmas()[0].antonyms())
```

```
Hypernyms:  [Synset('housing.n.01')]
Hyponyms:  [Synset('cliff_dwelling.n.01'), Synset('condominium.n.01'), Synset('fixer-upp
er.n.01'), Synset('hearth.n.02'), Synset('hermitage.n.01'), Synset('homestead.n.03'), Sy
nset('house.n.01'), Synset('lake_dwelling.n.01'), Synset('lodge.n.05'), Synset('messuag
e.n.01'), Synset('semi-detached_house.n.01'), Synset('vacation_home.n.01'), Synset('yur
t.n.01')]
Meronyms:  [Synset('bathroom.n.01'), Synset('bedroom.n.01'), Synset('den.n.04'), Synset
('dinette.n.01'), Synset('dining_room.n.01'), Synset('dressing_room.n.01'), Synset('fami
ly_room.n.01'), Synset('kitchen.n.01'), Synset('living_room.n.01')]
Holonynyms:  []
Antonyms:  []
```

In [ ]:
```
print(wn.synsets('swim'))

print("Definition: ",wn.synset('swim.v.01').definition())
print("Example of usage: ",wn.synset('swim.v.01').examples())
print("Lemmas: ",wn.synset('swim.v.01').lemmas())

hyper = lambda s: s.hypernyms()
swim = wn.synset('swim.v.01')
list(swim.closure(hyper))
```

```
[Synset('swimming.n.01'), Synset('swim.v.01'), Synset('float.v.02'), Synset('swim.v.0
3'), Synset('swim.v.04'), Synset('swim.v.05')]
Definition:  travel through water
Example of usage:  ['We had to swim for 20 minutes to reach the shore', 'a big fish was
swimming in the tank']
Lemmas:  [Lemma('swim.v.01.swim')]
```

Out[ ]:
```
[Synset('travel.v.01')]
```

Verb Organization in WordNet

The hierarchy is organized the same for verbs as it is for nouns except there's no uniform top level for verbs like there is for nouns. Nouns have "entity" to represent the top of the hierarchy but verbs don't have that. As seen in the above section it goes from swim to travel which makes sense in this context but if looked at with an example from class, dog, the verb goes to pursue then travel which is a little more of a stretch.

In [ ]:
```
print(wn.morphy('swimming', wn.VERB))
print(wn.morphy('swam', wn.VERB))
print(wn.morphy('swims', wn.VERB))
```

```
swim
swim
swim
```

In [ ]:
```
from nltk.wsd import lesk

hand =  wn.synset('hand.n.01')
foot = wn.synset('foot.n.01')
# wn similarity path
print(hand.path_similarity(foot))

# wu palmer similarity
print(wn.wup_similarity(hand,foot))

# lesk algorithm
sent = ['I','hurt', 'my','foot']
for ss in wn.synsets('foot'):
    print(ss, ss.definition())
print(lesk(sent,'foot'))
```

```
0.25
0.8235294117647058
Synset('foot.n.01') the part of the leg of a human being below the ankle joint
Synset('foot.n.02') a linear unit of length equal to 12 inches or a third of a yard
Synset('foot.n.03') the lower part of anything
Synset('animal_foot.n.01') the pedal extremity of vertebrates other than human beings
Synset('foundation.n.03') lowest support of a structure
Synset('foot.n.06') any of various organs of locomotion or attachment in invertebrates
Synset('foot.n.07') travel by walking
Synset('foot.n.08') a member of a surveillance team who works on foot or rides as a pass
enger
Synset('infantry.n.01') an army unit consisting of soldiers who fight on foot
Synset('metrical_foot.n.01') (prosody) a group of 2 or 3 syllables forming the basic uni
t of poetic rhythm
Synset('foot.n.11') a support resembling a pedal extremity
Synset('foot.v.01') pay for something
Synset('foot.v.02') walk
Synset('foot.v.03') add a column of numbers
Synset('infantry.n.01')
```

Observations:

The two words I wanted to fins the similarity of were 'hand' and 'foot'. When I did the regular nltk path similarity I only got .25 which is a lot lower than I was thinking or would expect. When I ran the Wu-Palmer similarity metric I got .82 which is more what I was expecting. The lesk algorithm was pretty interesting when I ran it for the word 'foot' and didn't give me what you would expect. I ran the sentence 'I hurt my foot' with the word 'foot' and got back 'infantry' from the lesk algorithm. The lesk algorithm looks at the context of the sentence around the word to try and find the best dictionary definition but in this case it didn't find the one I was thinking when i wrote the sentence.

SentiWordNet:

SentiWordNet can be imported in NLTK and it basically assigns w sentiment scores for each synset, either positive, negative, neutral. In other words it does sentiment analysis, it takes text and sees if its more positive/negative or neutral. It can be used when looking at costumer feedback or reviews.

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

senti_list = list(swn.senti_synsets('sadness'))
for i in senti_list:
    print(i)

sent = 'You are my best friend, I love you'
tokens = sent.split()

for t in tokens:
    syn_list = list(swn.senti_synsets(t))
    if syn_list:
        syn = syn_list[0]
        print(t," positive score: ", syn.pos_score(), " negative score:", syn.neg_score
```

```
<sadness.n.01: PosScore=0.0 NegScore=0.75>
<sadness.n.02: PosScore=0.0 NegScore=0.625>
<gloominess.n.03: PosScore=0.0 NegScore=0.875>
are  positive score:  0.0  negative score: 0.0
best  positive score:  0.25  negative score: 0.0
I  positive score:  0.0  negative score: 0.0
love  positive score:  0.625  negative score: 0.0
```

SentiWordNet Observations:

The scores I got back when using SentiWordNet seemed to be pretty reliable. I used the word sadness and got pretty high negative scores back with no positive scores which is accurate. The sentence I made up was 'You are my best friend, I love you' and the scores I got back leaned toward positive which again checks out with the sentence meaning. These scores are pretty useful to know in an NLP application because it allows you to get a number related to the positivity or negativity of a text.

Collocations:

A collocation is when two or more words occur together with a frequency greater than chance would suggest. One way to tell what is a collocation is when you can't substitute synonyms in and have it mean the same thing. Some common examples are 'crystal clear' and 'sick and tired'.

In [ ]:
```
from nltk.book import text4
import math

print(text4.collocations())

text = ' '.join(text4.tokens)
vocab = len(set(text4))
fc = text.count('Federal Government')/vocab
print("p(Federal Government) = ",fc)
f = text.count('Federal')/vocab
print("p(Federal) = ", f)
c = text.count('Government')/vocab
```

```
print('p(Government) = ', c)
pmi = math.log2(fc / (f * c))
print('pmi = ', pmi)
```

```
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
p(Federal Government) =  0.0031920199501246885
p(Federal) =  0.006483790523690773
p(Government) =  0.03371571072319202
pmi =  3.868067366919006
```

Collocations Observations:

One way to find a collocation is to use the point-wise mutual information, pmi, which was used in the code above. This formula takes the probability of x and y being together, divided by the probability of x times the probability of y. The pmi value can be positive meaning x and y occur together more than chance would allow or that it's likely a collocation, negative meaning that its not a collocation or a score of 0 which also means that they don't form a collocation. I looked at the pmi value of 'Federal Government' from text4 and got a value of 3.87 which tells us that it's likely a collocation.