AVP180003 Amol Perubhatla

WordNet is a large lexical database that contaions the semantic relations between the words of many languages. Parts of speech such as Nouns, verbs, and adjectives have been grouped into synsets. These synsets are linked through different relationships.

```
from nltk.metrics import BigramAssocMeasures
from nltk.collocations import BigramCollocationFinder
import itertools
from nltk.book import *
from nltk.collocations import BigramCollocationFinder, BigramAssocMeasures
import nltk
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
nltk.download('sentiwordnet')
from nltk.corpus import sentiwordnet as swn
from nltk.corpus import stopwords
```

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

```
print(wn.synsets('bear', pos=wn.NOUN))
bear = wn.synsets('bear', pos=wn.NOUN)[0]
```

```
[Synset('bear.n.01'), Synset('bear.n.02')]
```

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

```
massive plantigrade carnivorous or omnivorous mammals with long shaggy coats and strong
[]
[Lemma('bear.n.01.bear')]
Synset('carnivore.n.01')
Synset('placental.n.01')
Synset('mammal.n.01')
Synset('vertebrate.n.01')
Synset('chordate.n.01')
Synset('animal.n.01')
Synset('organism.n.01')
Synset('living_thing.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

The wordnet hierarchy for noun seems to get broader as you scale up the hierarchy. This is because the word falls under broader and broader categories as you scale

```
print(bear.hypernyms())
print(bear.hyponyms())
print(bear.part_meronyms())
print(bear.part_holonyms())
print(bear.lemmas()[0].antonyms())
```

```
[Synset('carnivore.n.01')]
[Synset('american_black_bear.n.01'), Synset('asiatic_black_bear.n.01'), Synset('bear_cub
[]
[]
[]
```

```
wn.synsets('walk', pos=wn.VERB)
walk = wn.synsets('walk', pos=wn.VERB)[0]
```

```
print(wn.synset('walk.v.01').definition())
print(wn.synset('walk.v.01').examples())
print(wn.synset('walk.v.01').lemmas())
hyp = walk.hypernyms()[0]
top = wn.synset('travel.v.01')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
use one's feet to advance; advance by steps
["Walk, don't run!", 'We walked instead of driving', 'She walks with a slight limp', 'Th
[Lemma('walk.v.01.walk')]
[Synset('travel.v.01')]
Synset('travel.v.01')
```

The hierarchy for verbs goes from narrower categories to a broader category at the top level.

```
wn.morphy('walk',wn.VERB)
wn.morphy('walking')
wn.morphy('walk', wn.VERB)
wn.morphy('walker', wn.VERB)
```

```
rabbit = wn.synsets('rabbit')[0]
deer = wn.synsets('deer')[0]
rabbit.path_similarity(deer)
```

```
0.125
```

```
wn.wup_similarity(rabbit,deer)
```

```
0.7586206896551724
```

```
for ss in wn.synsets('deer'):
    print(ss, ss.definition())
```

```
Synset('deer.n.01') distinguished from Bovidae by the male's having solid deciduous antl
```

```
sent = ['I','ate','the','deer']
print(lesk(sent,'deer'))
print(lesk(sent,'deer', pos='n'))
```

```
Synset('deer.n.01')
Synset('deer.n.01')
```

Because deer isn't as common a word as some others there aren't many synsets for it. Deer and rabbit have pretty high similarity because they are both forest animals. You can see this through the Wu Palmer similarity of .7586...

```
senti_list = list(swn.senti_synsets('sluggish'))
for item in senti_list:
    print(item)
```

```
<sluggish.s.01: PosScore=0.0 NegScore=0.0>
<dull.s.08: PosScore=0.0 NegScore=0.5>
<inert.s.03: PosScore=0.25 NegScore=0.125>
```

```
senti_list = list(swn.senti_synsets('sluggish','a'))
for item in senti_list:
    print(item)
```

```
<sluggish.s.01: PosScore=0.0 NegScore=0.0>
<dull.s.08: PosScore=0.0 NegScore=0.5>
<inert.s.03: PosScore=0.25 NegScore=0.125>
```

```
p = list(swn.senti_synsets('sluggish'))[0]
print("negative: ", p.neg_score())
print("positive: ", p.pos_score())
print("objective: ", p.obj_score())
```

```
negative:  0.0
positive:  0.0
objective:  1.0
```

```
sent = 'The terrible snail was sluggish as it returned home'
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]
        neg += syn.neg_score()
        pos += syn.pos_score()

print("neg\tpos counts")
print(neg, '\t', pos)
```

```
neg     pos counts
0.625    0.0
```

SentiWordNet can be used for sentiment analysis of words and sentences. It gives scores for objective, positive, and negative words in sentences. The polarity was negative because I added terrible to the sentence which has a negative sentiment. Before I added terrible the sentence was neutral as it was 0,0. These scores can be useful for deciding whether a sentence is positive or negative and can be used for understanding tone in text.

```
text4.collocations()
stopwords_ = set(stopwords.words('english'))
```

```
def bigram_word_feats(words, score_fn=BigramAssocMeasures.chi_sq, n=200):
    bigram_finder = BigramCollocationFinder.from_words(words)
    bigrams = bigram_finder.nbest(score_fn, n)
    return dict([(ngram, True) for ngram in itertools.chain(words, bigrams)
                if type(ngram) == tuple])
```

```
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
```

```
bigram_word_feats(text4)
```

```
{('/', '11'): True,
 ('25', 'straight'): True,
 ('Amelia', 'Island'): True,
 ('Apollo', 'astronauts'): True,
 ('Archibald', 'MacLeish'): True,
 ('BUSINESS', 'COOPERATION'): True,
 ('Barbary', 'Powers'): True,
 ('Belleau', 'Wood'): True,
 ('Boston', 'lawyer'): True,
 ('Britannic', 'Majesty'): True,
 ('COOPERATION', 'BY'): True,
 ('CRIMINAL', 'JUSTICE'): True,
 ('Calvin', 'Coolidge'): True,
 ('Cape', 'Horn'): True,
 ('Cardinal', 'Bernardin'): True,
 ('Chop', 'Hill'): True,
 ('Chosin', 'Reservoir'): True,
 ('Christmas', 'Eve'): True,
 ('Colonel', 'Goethals'): True,
 ('Dark', 'pictures'): True,
 ('Domestic', 'Product'): True,
 ('EIGHTEENTH', 'AMENDMENT'): True,
 ('Emancipation', 'Proclamation'): True,
 ('English', 'writer'): True,
 ('Fort', 'Sumter'): True,
 ('Gatun', 'dam'): True,
 ('Golden', 'Rule'): True,
 ('Gross', 'Domestic'): True,
 ('Growing', 'connections'): True,
 ('Hague', 'Tribunal'): True,
 ('Herein', 'flows'): True,
 ('Holy', 'Writ'): True,
 ('Hope', 'maketh'): True,
 ('Information', 'Age'): True,
 ('Iwo', 'Jima'): True,
 ('Joseph', 'Warren'): True,
 ('Julia', 'Coleman'): True,
 ('Khe', 'Sahn'): True,
 ('Lady', 'Michelle'): True,
     ('MANDATES', 'FROM'): True,
     ('Magna', 'Charta'): True,
     ('Mayflower', 'Compact'): True,
     ('Miss', 'Julia'): True,
     ('NATIONAL', 'INVESTIGATION'): True,
     ('Naval', 'Commissioners'): True,
     ('November', '1963'): True,
     ('OTHER', 'MANDATES'): True,
     ('Omaha', 'Beach'): True,
     ('PARTY', 'RESPONSIBILITIES'): True,
     ('PUBLIC', 'HEALTH'): True,
     ('Panama', 'Canal'): True,
     ('Penetrating', 'internally'): True,
     ('Persistent', 'importunity'): True,
     ('Philippine', 'Islands'): True,
     ('Pork', 'Chop'): True,
     ('Porto', 'Rico'): True,
     ('Reflecting', 'Pool'): True,
     ('Representative', 'Gillis'): True,
```

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

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

```
import math
vocab = len(set(text6))
hg = text.count('Citizens of')/vocab
print("p(Citizens of) = ", hg)
h = text.count('Citizens')/vocab
print("p(Citizens) = ", h)
g = text.count('of')/vocab
print('p(of) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
p(Citizens of) =  0.0013850415512465374
p(Citizens) =  0.003231763619575254
p(of) =  3.4653739612188366
pmi =  -3.0154034686150966
```

The probablity of these words is quite low, but citizens has a higher mutual information than of which shows that it is more likely to show up in the collocation.