Wordnet is developed around the concept of how humans hierarchically organize language. This includes nouns, verbs, adjectives, etc.

```
import nltk
nltk.download('gutenberg')
nltk.download('inaugural')
nltk.download('webtext')
nltk.download('genesis')
nltk.download('treebank')
nltk.download('stopwords')
nltk.download('nps_chat')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn
nltk.download('sentiwordnet')
from nltk.book import *
```

```
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Package webtext 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 treebank to /root/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
```

```
wn.synsets('house')
```

```
[Synset('house.n.01'),
 Synset('firm.n.01'),
 Synset('house.n.03'),
 Synset('house.n.04'),
 Synset('house.n.05'),
 Synset('house.n.06'),
 Synset('house.n.07'),
 Synset('sign_of_the_zodiac.n.01'),
```

```
 Synset('house.n.09'),
 Synset('family.n.01'),
 Synset('theater.n.01'),
 Synset('house.n.12'),
 Synset('house.v.01'),
 Synset('house.v.02')]
```

```
wn.synset('house.n.01').definition()
```

```
'a dwelling that serves as living quarters for one or more families'
```

```
wn.synset('house.n.01').examples()
```

```
['he has a house on Cape Cod', 'she felt she had to get out of the house']
```

```
wn.synset('house.n.01').lemmas()
```

```
[Lemma('house.n.01.house')]
```

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

```
 Synset('building.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')
```

It seems that at the top of every wordnet noun hierarchy is entity. There are many levels to the noun and get very specific like how whole is defined prio to object.

```
wn.synsets('sell')
```

```
[Synset('sell.n.01'),
 Synset('sell.v.01'),
 Synset('sell.v.02'),
 Synset('sell.v.03'),
 Synset('deal.v.06'),
 Synset('sell.v.05'),
 Synset('sell.v.06'),
```

```
 Synset('sell.v.07'),
 Synset('betray.v.02')]
```

```
wn.synset('sell.v.01').definition()
```

```
'exchange or deliver for money or its equivalent'
```

```
wn.synset('sell.v.01').examples()
```

```
['He sold his house in January',
 'She sells her body to survive and support her drug habit']
```

```
wn.synset('sell.v.01').lemmas()
```

```
[Lemma('sell.v.01.sell')]
```

```
wn.synsets('sell', 'v')[0].root_hypernyms()
```

```
[Synset('transfer.v.05')]
```

```
sell = wn.synset('sell.v.01')
hyp = sell.hypernyms()[0]
top = wn.synset('transfer.v.05')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
 Synset('exchange.v.01')
 Synset('transfer.v.05')
```

Verbs are organized less specifically than nouns in wordnet. There are only two levels above sell. What is interesting is how they decided that exchange is a lower level than transfer.

```
print(wn.morphy('sell'))
```

```
sell
```

```
print(wn.morphy('house'))
```

```
house
```

```
apartment = wn.synset('apartment.n.01')
```

```
condo = wn.synset('condominium.n.01')
wn.wup_similarity(apartment, condo)
```

```
0.8235294117647058
```

Based on the wp analysis, it shows that condo and apartment are very similar. This is a very accurate reading meanwhile lesk was less accurate. It found that the apartment is the closest similarity between condo and apartment.

```
print(lesk("condo", "apartment"))
```

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

SentiWordNet offers the functionality of checking whether a word is used in a positive, negative, or neutral way. This allows us to see if a sentence is considered to be intepreted in a good or bad way.

```
disingenuous = swn.senti_synset('disingenuous.a.01')
print(disingenuous)
```

```
<disingenuous.a.01: PosScore=0.0 NegScore=0.875>
```

```
sent = 'i did not like this show'
tokens = sent.split()
for token in tokens:
  syn_list = list(swn.senti_synsets(token))
  syn = syn_list[0]
  print("negative: ", syn.neg_score())
  print("positive: ", syn.pos_score())
  print("objective: ", syn.obj_score())
```

```
negative:  0.0
positive:  0.0
objective:  1.0
negative:  0.0
positive:  0.0
objective:  1.0
negative:  0.625
positive:  0.0
```

Wordnet did a good job showing that this sentence is a good mixture of objectice and negative. We can use this analysis to determine that this is a negative review of something.

Collocation is a string of two words meant to represent one singular thing.

```
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
```

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

```
p(United States) =  0.07340720221606649
p(United) =  0.07894736842105263
p(States) =  0.1528162511542013
pmi =  2.605160561199567
```

Double-click (or enter) to edit