WordNet is a lexical database of English that resembles a thesaurus providing glosses and relations to other words. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms called synsets. Synsets are interlinked by semantic and lexical relations.

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('sentiwordnet')
nltk.download('book')
```

```
from nltk.book import *
```

```
from nltk.corpus import wordnet as wn
print('Synsets: ', wn.synsets('rain'))
```

```
Synsets:  [Synset('rain.n.01'), Synset('rain.n.02'), Synset('rain.n.03'), Synset('rain.v
```

```
print('Definition: ' + wn.synset('rain.n.02').definition())
print('Examples: ', wn.synset('rain.n.02').examples())
print('Lemmas: ', wn.synset('rain.n.02').lemmas())

print('Hierarchy: ')
synset = wn.synset('rain.n.02')
while synset.hypernyms():
  print(synset.hypernyms())
  synset = synset.hypernyms()[0]
```

```
Definition: drops of fresh water that fall as precipitation from clouds
Examples:  []
Lemmas:  [Lemma('rain.n.02.rain'), Lemma('rain.n.02.rainwater')]
Hierarchy:
[Synset('fresh_water.n.01')]
[Synset('water.n.01')]
[Synset('binary_compound.n.01'), Synset('liquid.n.03')]
[Synset('compound.n.02')]
[Synset('chemical.n.01')]
[Synset('material.n.01')]
[Synset('substance.n.01')]
[Synset('matter.n.03'), Synset('part.n.01')]
[Synset('physical_entity.n.01')]
[Synset('entity.n.01')]
```

WordNet estalishes a hierarchy for relation between nouns. We can observe from my example that when we traverse up the hierarchy, the noun become more abstract or vague. Eventually, it eventually stops at the noun "entity". This "entity" noun serves as the root of all nouns in WordNet.

```
print('Hypernyms: ', wn.synset('rain.n.02').hypernyms())
print('Hyponyms: ', wn.synset('rain.n.02').hyponyms())
print('Meronyms: ', wn.synset('rain.n.02').part_meronyms())
print('Holonyms: ', wn.synset('rain.n.02').part_holonyms())
print('Antonyms: ', wn.synset('rain.n.02').lemmas()[0].antonyms())
```

```
Hypernyms:  [Synset('fresh_water.n.01')]
Hyponyms:  []
Meronyms:  []
Holonyms:  []
Antonyms:  []
```

```
print('Synsets: ', wn.synsets('stare'))
```

```
Synsets:  [Synset('stare.n.01'), Synset('gaze.v.01'), Synset('stare.v.02')]
```

```
print('Definition: ' + wn.synset('stare.v.02').definition())
print('Examples: ', wn.synset('stare.v.02').examples())
print('Lemmas: ', wn.synset('stare.v.02').lemmas())

print('Hierarchy: ')
synset = wn.synset('stare.v.02')
hyper = lambda s: s.hypernyms()
list(synset.closure(hyper))
```

```
Definition: fixate one's eyes
Examples:  ['The ancestor in the painting is staring down menacingly']
Lemmas:  [Lemma('stare.v.02.stare')]
Hierarchy: 
[Synset('look.v.01')]
```

From my observation, verbs are also organized into hierarchy. However, there is no root verb that can represent all verbs at the top of the hierarchy. In my example above, the top hierarchy for "stare" is "look" which cannot represent all actions/states/occurences.

```
print(wn.morphy('do'))
print(wn.morphy('did'))
print(wn.morphy('doing'))
print(wn.morphy('done'))
print(wn.morphy('does'))
```

```
do
do
do
do
doe
```

```
from nltk.wsd import lesk

tiger = wn.synset('tiger.n.02')
lion = wn.synset('lion.n.01')

print('Wu-Palmer: ', wn.wup_similarity(tiger, lion))
print("Lesk tiger's definition to lion: ", lesk(tiger.definition().split(), 'lion', 'n'))
print("Lesk lion's definition to tiger: ", lesk(lion.definition().split(), 'tiger', 'n'))
```

```
Wu-Palmer:  0.9333333333333333
Lesk tiger's definition to lion:  Synset('lion.n.01')
Lesk lion's definition to tiger:  Synset('tiger.n.02')
```

Since tiger and lion are from the same family in the animal kingdom, I thought it would have a high similarity. From my observation, running the Wu-Palmer similarity metric gave a 0.933 similarity which is nearly identical. Additionally, running the Lesk algorithm using the one's definition and other's noun returned the expected synset as well. This show that the 2 synsets I have chosen do indeed have a high similarity.

```
from nltk.corpus import sentiwordnet as swn

disrespect = swn.senti_synset('disrespect.n.01')
print(disrespect)
print("Positive score = ", disrespect.pos_score())
print("Negative score = ", disrespect.neg_score())
print("Objective score = ", disrespect.obj_score())

sent = '\nit is raining heavily but I like it for how calm and relaxing it is'
print(sent)
pos = 0
neg = 0
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        print(token, '\tPositive: ', syn.pos_score(), '\tNegative: ', syn.neg_score(), '\tObj
```

```
<disrespect.n.01: PosScore=0.0 NegScore=1.0>
Positive score =  0.0
Negative score =  1.0
Objective score =  0.0

it is raining heavily but I like it for how calm and relaxing it is
it      Positive:  0.0  Negative:  0.0  Objective:  1.0
is      Positive:  0.25         Negative:  0.125        Objective:  0.625
raining         Positive:  0.0  Negative:  0.125        Objective:  0.875
heavily         Positive:  0.0  Negative:  0.0  Objective:  1.0
but     Positive:  0.0  Negative:  0.0  Objective:  1.0
```

```
I       Positive:  0.0  Negative:  0.0  Objective:  1.0
like    Positive:  0.125        Negative:  0.0  Objective:  0.875
it      Positive:  0.0  Negative:  0.0  Objective:  1.0
calm    Positive:  0.375        Negative:  0.0  Objective:  0.625
relaxing        Positive:  0.0  Negative:  0.125        Objective:  0.875
it      Positive:  0.0  Negative:  0.0  Objective:  1.0
is      Positive:  0.25         Negative:  0.125        Objective:  0.625
```

The sentiment synsets does a good job at scoring a word as positive, negative, or objective. This could be really useful for performing sentiment analysis. Specifically, I think it could be used to determine whether a news article is factual or opinionated. It could be considered factual if the positive and negative score are balanced for every paragraph in the news article. Otherwise, the article is opinionated if the positive or negative score dominate the other.

Collocation is the frequent juxtaposition of two or more words that happens more often than chance. It is not possible to get the same meaning by substituting a word.

```
from nltk.probability import FreqDist
import math
text4.collocations()
print()

bgrams = nltk.bigrams(text4)
fdbig = nltk.FreqDist(bgrams)
fd = FreqDist(text4)
n_text4 = fd.N()
n_bgrams = fdbig.N()
n_public = fd['public']
n_debt = fd['debt']
n_publicdebt = fdbig[('public', 'debt')]

print("PMI for public debt: ", math.log((n_publicdebt/n_bgrams)/((n_public/n_text4)*(n_debt/n
```

```
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

PMI for public debt:  8.355999141055399
```

Following the PMI formula to verify the collocation in text4, I got a positive number much greater than 0 at 8.36. From the result, it is safe to claim that "public debt" is likely a collocation of text4. Because the higher the positive PMI is, the higher chance of the combined words being a collocation.