WordNet is a lexical database of words in multiple languages which have adjectives, adverbs, nouns, and verbs grouped differntly into a set of cognitive synonyms, where each word in the database is expressing its distinct concept. The cognitive synonyms which are called synsets are presented in the database with lexical and semantic relations. These words show the same concept of using them in similar contexts by interchanging them.

```
from nltk.corpus import wordnet as wn
```

```
wn.synsets('rat')
```

```
[Synset('rat.n.01'),
 Synset('scab.n.01'),
 Synset('rotter.n.01'),
 Synset('informer.n.01'),
 Synset('rat.n.05'),
 Synset('rat.v.01'),
 Synset('rat.v.02'),
 Synset('fink.v.01'),
 Synset('rat.v.04'),
 Synset('rat.v.05'),
 Synset('denounce.v.04')]
```

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

```
'a person who is deemed to be despicable or contemptible'
```

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

```
['only a rotter would do that',
 'kill the rat',
 'throw the bum out',
 'you cowardly little pukes!',
 "the British call a contemptible person a `git'"]
```

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

```
[Lemma('rotter.n.01.rotter'),
 Lemma('rotter.n.01.dirty_dog'),
 Lemma('rotter.n.01.rat'),
 Lemma('rotter.n.01.skunk'),
 Lemma('rotter.n.01.stinker'),
 Lemma('rotter.n.01.stinkpot'),
 Lemma('rotter.n.01.bum'),
 Lemma('rotter.n.01.puke'),
 Lemma('rotter.n.01.crumb'),
 Lemma('rotter.n.01.lowlife'),
 Lemma('rotter.n.01.scum_bag'),
 Lemma('rotter.n.01.so-and-so'),
 Lemma('rotter.n.01.git')]
```

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

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

```
Synset('unpleasant_person.n.01')
Synset('unwelcome_person.n.01')
Synset('person.n.01')
Synset('causal_agent.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

This while loop tranverses up the hierarchy of nouns since entity is the top of the list and unpleasant person is at the lowest, we can see that there isn't too many nouns that can be listed for the word rotter. Everytime we iterate through the loop if the word entity isn't the same as the incoming hypernym then we print the hypernym and continue until we reach the word entity and then the program breaks meaning to stop running the program. We only hypernyms in this instance since we are traversing up the ladder. If we were to traverse down the ladder then we would have used hyponym to go the opposite direction going from highest to lowest starting from entity and ending in unpleasant person.

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

```
[Synset('unpleasant_person.n.01')]
[]
[]
[]
[]
```

```
wn.synsets('running')
```

```
[Synset('run.n.05'),
 Synset('run.n.07'),
 Synset('running.n.03'),
 Synset('running.n.04'),
 Synset('track.n.11'),
 Synset('run.v.01'),
 Synset('scat.v.01'),
 Synset('run.v.03'),
 Synset('operate.v.01'),
 Synset('run.v.05'),
 Synset('run.v.06'),
 Synset('function.v.01'),
 Synset('range.v.01'),
 Synset('campaign.v.01'),
 Synset('play.v.18'),
 Synset('run.v.11'),
 Synset('tend.v.01'),
 Synset('run.v.13'),
 Synset('run.v.14'),
 Synset('run.v.15'),
 Synset('run.v.16'),
 Synset('prevail.v.03'),
 Synset('run.v.18'),
 Synset('run.v.19'),
 Synset('carry.v.15'),
 Synset('run.v.21'),
 Synset('guide.v.05'),
 Synset('run.v.23'),
 Synset('run.v.24'),
 Synset('run.v.25'),
 Synset('run.v.26'),
 Synset('run.v.27'),
 Synset('run.v.28'),
 Synset('run.v.29'),
 Synset('run.v.30'),
 Synset('run.v.31'),
 Synset('run.v.32'),
 Synset('run.v.33'),
 Synset('run.v.34'),
 Synset('ply.v.03'),
 Synset('hunt.v.01'),
 Synset('race.v.02'),
 Synset('move.v.13'),
 Synset('melt.v.01'),
 Synset('ladder.v.01'),
 Synset('run.v.41'),
 Synset('running.a.01'),
 Synset('running.s.02'),
 Synset('running.a.03'),
 Synset('running.a.04'),
 Synset('linear.s.05'),
 Synset('running.s.06')]
```

```
wn.synset('run.v.03').definition()
```

```
'stretch out over a distance, space, time, or scope; run or extend between two points or
ain point'
```

```
wn.synset('run.v.03').examples()
```

```
['Service runs all the way to Cranbury',
 "His knowledge doesn't go very far",
 'My memory extends back to my fourth year of life',
 'The facts extend beyond a consideration of her personal assets']
```

```
wn.synset('run.v.03').lemmas()
```

```
[Lemma('run.v.03.run'),
 Lemma('run.v.03.go'),
 Lemma('run.v.03.pass'),
 Lemma('run.v.03.lead'),
 Lemma('run.v.03.extend')]
```

```
run = wn.synset('run.v.03')
```

```
hyper = lambda s: s.hypernyms()
list(run.closure(hyper))
```

```
[Synset('be.v.03')]
```

When trying to tranverse through hypernyms of the verb "run", there is no top level uniform synset so some words be infinitely tranverse forever. In order to read a properly and easier, we can use lambda function to run the closure of the verb run which this method is also useful for traversing through noun synsets. In this instance we can see that the verb "be" is the only word that is printed which means that there is no hierarchy.

```
print(wn.morphy("run", wn.VERB))
print(wn.morphy("running", wn.VERB))
print(wn.morphy("runs", wn.VERB))
print(wn.morphy("ran", wn.VERB))
print(wn.morphy("running", wn.VERB))
```

```
run
run
run
run
run
```

```
ocean = wn.synset('ocean.n.01')
sea = wn.synset('sea.n.01')
```

```
ocean.wup_similarity(sea)
```

```
0.8
```

```
ocean.path_similarity(sea)
```

```
0.3333333333333333
```

```
from nltk.wsd import lesk
sent2 = ['The','ocean','is','blue']
print(lesk(sent2, 'ocean'))
print()
for ss in wn.synsets('ocean'):
    print(ss, ss.definition())
```

```
Synset('ocean.n.02')

Synset('ocean.n.01') a large body of water constituting a principal part of the hydrosph
Synset('ocean.n.02') anything apparently limitless in quantity or volume
```

What I found interesting is that the sentence that I used to found the synset for 'ocean.n.02' has a wup_similarity rating of .18 in comparison with the word 'sea' while the synset I was using originall, ocean.n.01, has the score of 0.8. For the definition of ocean.n.02, it reads that it's a measurement of volume but what I was hoping for was ocean.n.01 which is the correct definition of ocean that I was looking for.

SentiWordNet is about picking a word on your own or using a word from a sentence and there's a measurement of how positive or negative that word may be. The word can be positive depending on the context of the sentence and how the word is used in the sentence. SentiWordNet can be used to prove whether a word in certain context is positive, negative, or neutral in some cases.

```
from nltk.corpus import sentiwordnet as swn

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

```
<damaging.s.01: PosScore=0.0 NegScore=0.75>
```

```
sent = 'My mother hates me very much because I cheated on my math test.'
```

```
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    for item in syn_list:
```

```
    print(item)
```

```
<mother.n.01: PosScore=0.0 NegScore=0.0>
<mother.n.02: PosScore=0.0 NegScore=0.0>
<mother.n.03: PosScore=0.0 NegScore=0.0>
<mother.n.04: PosScore=0.0 NegScore=0.0>
<mother.n.05: PosScore=0.0 NegScore=0.0>
<mother.v.01: PosScore=0.0 NegScore=0.0>
<beget.v.01: PosScore=0.0 NegScore=0.0>
<hate.n.01: PosScore=0.125 NegScore=0.375>
<hate.v.01: PosScore=0.0 NegScore=0.75>
<hat.v.01: PosScore=0.0 NegScore=0.0>
<hat.v.02: PosScore=0.0 NegScore=0.0>
<maine.n.01: PosScore=0.0 NegScore=0.0>
<very.s.01: PosScore=0.5 NegScore=0.0>
<identical.s.02: PosScore=0.5 NegScore=0.125>
<very.r.01: PosScore=0.25 NegScore=0.25>
<very.r.02: PosScore=0.25 NegScore=0.0>
<much.n.01: PosScore=0.125 NegScore=0.125>
<much.a.01: PosScore=0.0 NegScore=0.0>
<much.r.01: PosScore=0.125 NegScore=0.0>
<much.r.02: PosScore=0.125 NegScore=0.0>
<a_lot.r.01: PosScore=0.25 NegScore=0.0>
<much.r.04: PosScore=0.125 NegScore=0.125>
<much.r.05: PosScore=0.375 NegScore=0.0>
<iodine.n.01: PosScore=0.0 NegScore=0.0>
<one.n.01: PosScore=0.0 NegScore=0.0>
<i.n.03: PosScore=0.0 NegScore=0.0>
<one.s.01: PosScore=0.0 NegScore=0.25>
<cheat.v.01: PosScore=0.0 NegScore=0.0>
<cheat.v.02: PosScore=0.0 NegScore=0.0>
<cheat.v.03: PosScore=0.0 NegScore=0.125>
<cheat_on.v.01: PosScore=0.0 NegScore=0.5>
<on.a.01: PosScore=0.0 NegScore=0.0>
<on.a.02: PosScore=0.0 NegScore=0.0>
<along.r.01: PosScore=0.0 NegScore=0.0>
<on.r.02: PosScore=0.125 NegScore=0.0>
<on.r.03: PosScore=0.0 NegScore=0.0>
<mathematics.n.01: PosScore=0.0 NegScore=0.0>
```

I noticed that some of the nouns and adjectives in the sentence didn't have a polarity score which makes sense because nouns don't really have some sort of emotion to it unless it's under certain context. Generally, adjectives don't have a score. The scores are a good representation of how AI can learn that if there is a higher score for negativity, then you can expect that person who wrote the sentence to be upset.

When two or more words usually occur together with a frequency greater than chance would suggest, the words may form a collocation. Collocations use two words that work well together and are familiar to recognize.

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

```
    *** Introductory Examples for the NLTK Book ***
    Loading text1, ..., text9 and sent1, ..., sent9
    Type the name of the text or sentence to view it.
    Type: 'texts()' or 'sents()' to list the materials.
    text1: Moby Dick by Herman Melville 1851
    text2: Sense and Sensibility by Jane Austen 1811
    text3: The Book of Genesis
    text4: Inaugural Address Corpus
    text5: Chat Corpus
    text6: Monty Python and the Holy Grail
    text7: Wall Street Journal
    text8: Personals Corpus
    text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

```
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]

import math
vocab = len(set(text4))
hg = text.count('United States')/vocab
print("p(United States) = ",hg )
h = text.count('United')/vocab
print("p(United States) = ", h)
g = text.count('States')/vocab
print('p(States) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
    p(United States) =  0.015860349127182045
    p(United States) =  0.0170573566084788
    p(States) =  0.03301745635910224
    pmi =  4.815657649820885
```

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

import math
vocab = len(set(text4))
hg = text.count('Vice President')/vocab
print("p(Vice President) = ",hg )
h = text.count('Vice')/vocab
print("p(Vice) = ", h)
```

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

```
p(Vice President) =  0.0017955112219451373
p(Vice) =  0.0018952618453865336
p(President) =  0.010773067331670824
pmi =  6.458424602064904
```

The results show that the pmi for 'United Stat
President' which means that 'United States' ha
than 'Vice President'.

The results show that the pmi for 'United States' is higher than 'Vice President' which means that 'United States' has a better mutual information than 'Vice President'.