WordNet is a hierarchical organization of nouns, verbs adjectives, and adverbs including their glossary, set of synonyms, examples, and relationships with other words. It contains relationships between words for words in more than 200 languages. It links words into hierarchical relationships using synonyms, hyponyms, meronyms, and hypernyms.

Noun Chosen: "cake"

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('gutenberg')
nltk.download('sentiwordnet')
nltk.download('genesis')
nltk.download('webtext')
nltk.download('nps_chat')
nltk.download('treebank')
nltk.download('inaugural')
```

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

The way synset is organized for nouns is in a hierarchical manner of highly connected synsets. Nouns have the top set to 'entity' which makes it possible to traverse the hierarchy of the synsets. One way to traverse up of the hierarchy is to recursively call hypernyms until we reach the top entity.

```
from nltk.corpus import wordnet as wn
noun = "cake"
synsets = wn.synsets(noun)
synset = synsets[0]
```

```
print(synsets)

print("Synset chosen is ", synset )
print(synset.definition())
print(synset.examples())
print(synset.lemmas())

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

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

```
[Synset('cake.n.01'), Synset('patty.n.01'), Synset('cake.n.03'), Synset('coat.v.
Synset chosen is  Synset('cake.n.01')
a block of solid substance (such as soap or wax)
['a bar of chocolate']
[Lemma('cake.n.01.cake'), Lemma('cake.n.01.bar')]
Synset('block.n.01')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
[Synset('block.n.01')]
[Synset('tablet.n.03')]
[]
[]
[]
```

```
verb = "walk"
synsets = wn.synsets(verb)
synset = synsets[0]
print(synsets)

print("Synset chosen is ", synset )
print(synset.definition())
print(synset.examples())
print(synset.lemmas())

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

```
[Synset('walk.n.01'), Synset('base_on_balls.n.01'), Synset('walk.n.03'), Synset(
Synset chosen is  Synset('walk.n.01')
the act of traveling by foot
['walking is a healthy form of exercise']
[Lemma('walk.n.01.walk'), Lemma('walk.n.01.walking')]
[Synset('locomotion.n.02'),
 Synset('motion.n.06'),
 Synset('change.n.03'),
 Synset('action.n.01'),
 Synset('act.n.02'),
 Synset('event.n.01'),
 Synset('psychological_feature.n.01'),
 Synset('abstraction.n.06'),
 Synset('entity.n.01')]
```

Verbs are organized in hypernym/hyponym relationship. Unlike nouns, they do not have a top so it is difficult to traverse in the same way as nouns. A better approach is using the closure method provided by nltk to pass in a lambda function.

Double-click (or enter) to edit

```
morphy_verb = wn.morphy(verb)
print(wn.synsets(morphy_verb))
```

```
[Synset('walk.n.01'), Synset('base_on_balls.n.01'), Synset('walk.n.03'), Synset(
```

```
word_one = "help"
word_two = "assist"


word_one_syn = wn.synsets(word_one)
word_two_syn = wn.synsets(word_two)
print(word_one_syn)
print(word_two_syn)
wup = wn.wup_similarity(word_one_syn[0], word_two_syn[0])
print("Wu_Palmer Similarity: ", wup)

from nltk.wsd import lesk
```

```
sent = ['I', 'am' , 'angry', '.']
print(lesk(sent, "angry"))
```

```
[Synset('aid.n.02'), Synset('assistant.n.01'), Synset('aid.n.01'), Synset('avail
[Synset('aid.n.02'), Synset('assist.n.02'), Synset('help.v.01'), Synset('assist.
Wu_Palmer Similarity:  1.0
Synset('angry.s.03')
```

From my observation I observed that the reason that the WUP index is one is because if we print the hierarchy for both help and assist they are only one level up before they share the same synset ('aid.n.02') and since the WUP looks at common path it makes sense that the index is one.

The SentiWordNet is used to analyze bodies of text in order to determine if it positive/negative/or subjective. Additionally, it also provides information on the degree in which a certain word displays the aforementioned characteristics. It has various practical applications such as analyzing journals to analyze the participants emotions, to flag inappropriate and vulgar text on social media website automatically and so on.

```
from nltk.corpus import sentiwordnet as swn
word = "sad"
senti_list = list(swn.senti_synsets(word))
for item in senti_list:
    print(item)
sent = ['I', 'am' , 'angry', '.']

for a_word in sent:
    print("For the word: ", a_word)
    senti_list = list(swn.senti_synsets(a_word))
    if senti_list:
      for item in senti_list:
        print(item)
      print()
```

```
<sad.a.01: PosScore=0.125 NegScore=0.75>
<sad.s.02: PosScore=0.0 NegScore=0.25>
<deplorable.s.01: PosScore=0.0 NegScore=1.0>
For the word:  I
<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>

For the word:  am
```

```
<americium.n.01: PosScore=0.0 NegScore=0.0>
<master_of_arts.n.01: PosScore=0.0 NegScore=0.125>
<amplitude_modulation.n.01: PosScore=0.0 NegScore=0.0>
<be.v.01: PosScore=0.25 NegScore=0.125>
<be.v.02: PosScore=0.0 NegScore=0.0>
<be.v.03: PosScore=0.0 NegScore=0.0>
<exist.v.01: PosScore=0.0 NegScore=0.0>
<be.v.05: PosScore=0.0 NegScore=0.0>
<equal.v.01: PosScore=0.125 NegScore=0.125>
<constitute.v.01: PosScore=0.0 NegScore=0.0>
<be.v.08: PosScore=0.0 NegScore=0.0>
<embody.v.02: PosScore=0.0 NegScore=0.0>
<be.v.10: PosScore=0.0 NegScore=0.0>
<be.v.11: PosScore=0.0 NegScore=0.0>
<be.v.12: PosScore=0.0 NegScore=0.0>
<cost.v.01: PosScore=0.0 NegScore=0.0>

For the word:  angry
<angry.a.01: PosScore=0.375 NegScore=0.375>
<angry.s.02: PosScore=0.375 NegScore=0.5>
<angry.s.03: PosScore=0.0 NegScore=0.875>

For the word:  .
```

The scores here show show whether the words have a positive or negative score and in addition shows the degree to which they are negative or positive. This can have real life applications especially if we are trying to automatically flag profance tweets or other social media posts by seeing the ratio between pos to neg words, the fraction of negative words in the text, and so on to make an informed decision to estimate if a text body is malicious or not.

A collocation is when two or more words are combined to form a phrase that cannot be substituted by a single word to yield the same meaning. One good example of this is United States, or fellow citizen

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

text = ' '.join(text4.tokens)
vocab = len(set(text4))
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)
```

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

The results show a positive pmi and rather high positive pmi which suggests that "United States" is most likely a collocation.