Summary of WordNet

WordNet is a database of nouns, verbs, adjectives and their relationships, linking them together by their symantic relationships and grouping them into synonym sets called synsets. WordNet is stored in hierarchical relationships to support the theory that humans mentally group concepts in a hierarchial structure. WordNet's use cases include word sense disambiguation, sentiment analysis, and finding word similarities.

```
import nltk
nltk.download('popular')
nltk.download('sentiwordnet')
nltk.download('book')
from nltk.book import text4
from nltk.corpus import wordnet as wn
```

```
noun = wn.synsets('dog', pos=wn.NOUN)
print(noun)
```

```
[Synset('dog.n.01'), Synset('frump.n.01'), Synset('dog.n.03'), Synset('cad.n.01'), Sy
```

```
noun = noun[0]
noun_definition = noun.definition()
print("Definition:\t",noun_definition)
noun_examples = noun.examples()
print("Examples:\t", noun_examples)
noun_lemmas = noun.lemmas()
print("Lemmas:\t", noun_lemmas)

print("\nNoun hierarchy:")
hyp = noun.hypernyms()[0]
top = wn.synset('entity.n.01')
while hyp:
  print(hyp)
  if hyp == top:
    break
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
Definition:      a member of the genus Canis (probably descended from the common wolf
Examples:        ['the dog barked all night']
Lemmas:  [Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'), Lemma('dog.n.01.Cani

Noun hierarchy:
Synset('canine.n.02')
Synset('carnivore.n.01')
```

✓ 0s completed at 6:19 PM

```
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')
```

```
print("Hypernyms:\t", noun.hypernyms())
print("Hyponyms:\t", noun.hyponyms())
print("Meronyms:\t", noun.part_meronyms() + noun.substance_meronyms())
print("Holonyms:\t", noun.part_holonyms() + noun.substance_holonyms())
print("Antonyms:\t",noun.lemmas()[0].antonyms())
```

```
Hypernyms:      [Synset('canine.n.02'), Synset('domestic_animal.n.01')]
Hyponyms:       [Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.01'), S
Meronyms:       [Synset('flag.n.07')]
Holonyms:       []
Antonyms:       []
```

```
verb = wn.synsets('walk', pos=wn.VERB)
print(verb)
```

```
[Synset('walk.v.01'), Synset('walk.v.02'), Synset('walk.v.03'), Synset('walk.v.04'),
```

```
verb = verb[0]
verb_definition = verb.definition()
print("Definition:\t",verb_definition)
verb_examples = verb.examples()
print("Examples:\t",verb_examples)
verb_lemmas = verb.lemmas()
print("Lemmas:\t",verb_lemmas)

print('Verb hierarchy:')
hyper = lambda s: s.hypernyms()
print(list(verb.closure(hyper)))
```

```
Definition:     use one's feet to advance; advance by steps
Examples:       ["Walk, don't run!", 'We walked instead of driving', 'She walks with
Lemmas:  [Lemma('walk.v.01.walk')]
Verb hierarchy:
[Synset('travel.v.01')]
```

Verbs in WordNet

Unlike nouns, there is no uniform top level synset for verbs. To traverse the hierarchy, you will need to use a closure or find the root hypernym to know when the top of the hierarchy is reached.

```
print(wn.morphy('love'))
print(wn.morphy('love', wn.ADJ))
print(wn.morphy('love', wn.NOUN))
print(wn.morphy('love', wn.ADV))
print(wn.morphy('love', wn.VERB))
```

```
love
None
love
None
love
```

```
from nltk.wsd import lesk
word1 = wn.synsets('love')[0]
word2 = wn.synsets('infatuate')[0]

print(wn.wup_similarity(word1,word2))
print(lesk('He loved her for 2 years.'.split(), 'infatuate'))
```

```
0.16666666666666666
Synset('infatuate.v.01')
```

Lesk and Wu-Palmer Observations

I observed that while the words do have similar meaning, the Wu-Palmer metric returns a lower score than expected. For Lesk, I observed when using the similar word in the sentence, the Lesk algorithm was able to select the correct form of the original word.

SentiWordNet

SentiWordNet is a sentiment analysis tool built on top of WordNet that provides sentiment scores for a synset, positive, negative and objectivity. When provided a sysnet, SentiWordNet provides a score, for larger texts we can iterate over the tokens and add up each score to determine the overall sentiment of the text. Some use cases for SentiWordNet is to classify positive and negative reviews for a product, stock forecasts, and tracking sentiment for a product over time.

```
from nltk.corpus import sentiwordnet as swn
flourish = wn.synsets('flourish')[0]
```

```
senti_list = list(swn.senti_synsets('flourish'))

for item in senti_list:
  print(item)
```

```
     <flourish.n.01: PosScore=0.125 NegScore=0.0>
     <flourish.n.02: PosScore=0.0 NegScore=0.0>
     <flourish.n.03: PosScore=0.0 NegScore=0.0>
     <flourish.n.04: PosScore=0.0 NegScore=0.0>
     <flourish.n.05: PosScore=0.0 NegScore=0.0>
     <boom.v.05: PosScore=0.125 NegScore=0.0>
     <thrive.v.02: PosScore=0.125 NegScore=0.0>
     <brandish.v.01: PosScore=0.0 NegScore=0.0>
```

```
sentence = nltk.word_tokenize("The movie that we just saw was pretty bad.")
neg = 0
pos = 0
for token in sentence:
  syn_list = list(swn.senti_synsets(token))
  if syn_list:
    syn = syn_list[0]
    neg += syn.neg_score()
    pos += syn.pos_score()
print("Negative: " + str(neg))
print("Positive: " + str(pos))
```

```
     Negative: 1.0
     Positive: 1.5
```

Observation of SentiWordNet

Through my testing with the above sentence, I observed that SentiWordNet has some issues with sentences. Above, I used "pretty bad" in the sentence which has a negative connotation. However, SentiWordNet still scored the sentence as overall positive due to the word "pretty" being included. Still, I believe that having these scores greatly helps NLP applications that have to do with sentiment analysis though some caution has to be applied when using harder sentences to score like the one above.

Collocations

A collocation is a set of words that combine more that expected by chance. When they occur together, there is no single word that can be substituted and have the same meaning.

```
import math
text4.collocations()

text = ' '.join(text4.tokens)
vocab = len(set(text4))
us = text.count('United States') / vocab
print("\np(United States) = ", us)
u = text.count('United') / vocab
print("p(United) = ", u)
s = text.count('States') / vocab
print("p(States) = ", s)
pmi = math.log2(us / (u * s))
print("pmi = ", pmi)
```

```
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

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

PMI Results

Through calculating the PMI of "United States" I found that the collocation yielded a high score which means that United States is more likely to be a collocation. This is due to United States being a proper noun and a name for the country, and being a document related to the United States makes it more likely to appear together as well.