What is Wordnet?

Wordnet is a lexical database of semantic relationships between words. Words are organized by meaning and Wordnet can be very useful for NLP applications.

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

from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn

from nltk.book import *
text4
```

```
[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!
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg 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 inaugural to /root/nltk_data...
[nltk_data]   Package inaugural 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 webtext to /root/nltk_data...
[nltk_data]   Package webtext 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]   Package stopwords is already up-to-date!
<Text: Inaugural Address Corpus>
```

Noun

```
print(wn.synsets('qualm'))
```

```
[Synset('scruple.n.02'), Synset('queasiness.n.01')]
```

```
print(wn.synset('scruple.n.02').definition())

print(wn.synset('scruple.n.02').examples())

print(wn.synset('scruple.n.02').lemmas())
```

```
uneasiness about the fitness of an action
[]
[Lemma('scruple.n.02.scruple'), Lemma('scruple.n.02.qualm'), Lemma('scruple.n.02.misgiving')]
```

Hierarchy

```
hyp = wn.synset('scruple.n.02').hypernyms()[0]
top = wn.synset('entity.n.01')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
Synset('anxiety.n.02')
Synset('emotion.n.01')
Synset('feeling.n.01')
Synset('state.n.02')
Synset('attribute.n.02')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

It seems that hierarchy for nouns goes up by the broadness of the concept. Above we can see that anxiety and emotion pop up, anxiety is an emotion and even further that emotions are feelings. Eventually everything is an entity, so it is at the top of the hierarchy for nouns.

```
# hypernyms, hyponyms, meronyms, holonyms, antonyms
scruple = wn.synset('scruple.n.02')

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

```
[Synset('anxiety.n.02')]
[]
[]
[]
[]
```

Verb

```
print(wn.synsets('discombobulate'))
```

```
[Synset('bewilder.v.02'), Synset('confuse.v.02')]
```

```
print(wn.synset('bewilder.v.02').definition())

print(wn.synset('bewilder.v.02').examples())

print(wn.synset('bewilder.v.02').lemmas())
```

```
cause to be confused emotionally
[]
[Lemma('bewilder.v.02.bewilder'), Lemma('bewilder.v.02.bemuse'), Lemma('bewilder.v.02.discombobulate'), Lemma('bewilder.v.02.throw')]
```

```
bewilder = wn.synset('bewilder.v.02')

hyp = bewilder.hypernyms()[0]
top = bewilder.root_hypernyms()[0]
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
Synset('upset.v.02')
Synset('arouse.v.01')
Synset('make.v.03')
```

The organization for verbs is not as clear cut as the nouns above, but it looks like it still follows a similar concept of words encapsulating words below them in the hierarchy. Although, this looks to be more about similarity than it did for nouns. Upset could encapsulate bewilder, but past that it's less clear.

```
# bewilder, bewilders, bewildered, bewildering

print(wn.morphy('bewilders'))
print(wn.morphy('bewildered'))
print(wn.morphy('bewildering'))

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

```
bewilder
bewilder
bewilder
None
bewilder
None
None
```

Similarity between 2 words

Wu-Palmer

```
sever = wn.synset('sever.v.01')
cut = wn.synset('cut.v.01')
print(wn.wup_similarity(sever, cut))
```

```
0.6666666666666666
```

Unsurprisingly, the Wu-Palmer similarity metric found cut and sever to be pretty closely related. You could consider severing something to a type of cut after all.

Lesk

```
sentence = ['So',',','after','hot-wiring','his','car','she','severed','all','his', 'limbs', '.']
print(lesk(sentence, 'sever'))

sentence2 = ['She','then','realized','she','was', 'supposed','slice','a','watermelon','not', 'cut', 'a', 'fellow', 'in', 'half', '.']
print(lesk(sentence2, 'cut'))
```

```
Synset('sever.v.01')
Synset('reduce.v.01')
```

For the first sentence I expected the use of sever to be 'sever.v.01', however I didn't expect that 'reduce.v.01' would be found for the second sentence. Although it works, I felt 'cut.v.01' would've been more appropriate.

SentiWordNet

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

```
<horrid.s.01: PosScore=0.0 NegScore=0.875>
<hideous.s.01: PosScore=0.0 NegScore=0.875>
```

```
sent = "I can't believe he would do something so toxic"
neg = 0
pos = 0
tokens = sent.split()
for token in tokens:
    senti_list = list(swn.senti_synsets(token))
    if senti_list:
      print(senti_list[0])
    else:
      print("no sysnet")
```

```
<iodine.n.01: PosScore=0.0 NegScore=0.0>
no sysnet
<believe.v.01: PosScore=0.125 NegScore=0.0>
<helium.n.01: PosScore=0.0 NegScore=0.0>
no sysnet
<bash.n.02: PosScore=0.0 NegScore=0.0>
no sysnet
<sol.n.03: PosScore=0.0 NegScore=0.0>
<toxic.a.01: PosScore=0.0 NegScore=0.25>
```

SentiWordNet is used to determine the sentiment of a word, whether positive, negative, or neutral. Sentiment analysis could be used to monitor public opinion or identify communities with a lot of toxicity.

The sentence I analyzed above shows there was one postively charged word and one negatively charged word, but overall has a slight negative sentiment. Knowing these scores in an NLP application is useful because it gives us a general feeling the sentence evokes. This is again useful for monitoring public opinion.

Collocation

A collocation can be described a predicatable combination of words, words that usually appear with another. They form a meaning greater than the sum of their parts.

```
# get collocations
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
```

Mutual Information

```
text = ' '.join(text4.tokens)

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) = ", 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) =  0.0170573566084788
    p(States) =  0.03301745635910224
    pmi =  4.815657649820885
```

Unsurprisingly, United States has a high mutual information so it's very likely that it's a collocation. Intuitvely this makes sense since we usually see the words united and states together to mean the country.

✓ 0s completed at 5:33 PM