WordNet

WordNet is a lexical database that serves as an English dictionary meant for NLP. It links words by semantic relationship and can be used for translation, sentiment analysis, information retrieval, and more.

```
import·nltk
nltk.download("popular")
from nltk.corpus import wordnet as wn
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/shakespeare.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[nltk_data]    | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/words.zip.
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection popular
```

Nouns

```
# Synsets for noun "spring"
wn.synsets('spring')
```

```
[Synset('spring.n.01'),
 Synset('spring.n.02'),
 Synset('spring.n.03'),
 Synset('spring.n.04'),
 Synset('give.n.01'),
 Synset('leap.n.01'),
 Synset('jump.v.01'),
 Synset('form.v.03'),
 Synset('bounce.v.01'),
 Synset('spring.v.04'),
 Synset('spring.v.05')]
```

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

```
'the season of growth'
```

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

```
['the emerging buds were a sure sign of spring',
 'he will hold office until the spring of next year']
```

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

```
[Lemma('spring.n.01.spring'), Lemma('spring.n.01.springtime')]
```

```
print(wn.synset('spring.n.01').root_hypernyms())
```

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

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

```
Synset('season.n.02')
Synset('time_period.n.01')
Synset('fundamental_quantity.n.01')
Synset('measure.n.02')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

Through observation, we can see that WordNet organizes nouns through hierarchical semantic relations. These relations include hyponyms, meronyms, troponyms, hypernyms, and holonyms. To use our noun as an example, "season" is a hypernym of "spring" because it is a step less specific. Traversing up the hierarchy for a noun will always bring you to root hypernym, entity.

```
#Printing hypernyms, hyponyms, meronyms,holonyms, and antonyms.
spring = wn.synset('spring.n.01')

print('hypernyms: ', spring.hypernyms())
print('hyponyms: ', spring.hyponyms())
print('meronyms: ', spring.part_meronyms())
print('meronyms: ', spring.substance_meronyms())
print('holonyms: ', spring.part_holonyms())
print('holonyms: ', spring.substance_holonyms())

antonyms = []
for syn in wn.synsets("spring"):
    for i in syn.lemmas():
         if i.antonyms():
              antonyms.append(i.antonyms()[0].name())

if antonyms == []:
  print("antonyms: []")
else:
  print("antonyms: ", set(antonyms))
```

```
hypernyms:  [Synset('season.n.02')]
hyponyms:  []
meronyms:  [Synset('vernal_equinox.n.01')]
meronyms:  []
holonyms:  []
holonyms:  []
antonyms: []
```

Verbs

```
# Synsets for noun "grow"
wn.synsets('grow')
```

```
[Synset('turn.v.07'),
 Synset('grow.v.02'),
 Synset('grow.v.03'),
 Synset('grow.v.04'),
 Synset('mature.v.01'),
 Synset('originate.v.01'),
 Synset('grow.v.07'),
 Synset('grow.v.08'),
 Synset('develop.v.14'),
 Synset('grow.v.10')]
```

```
wn.synset('grow.v.02').definition()
```

```
'become larger, greater, or bigger; expand or gain'
```

```
wn.synset('grow.v.02').examples()
```

```
['The problem grew too large for me', 'Her business grew fast']
```

```
wn.synset('grow.v.02').lemmas()
```

```
[Lemma('grow.v.02.grow')]
```

```
print('root hypernyms: ', wn.synset('grow.v.02').root_hypernyms())
```

```
root hypernyms:  [Synset('change.v.02')]
```

```
hyp = wn.synset('grow.v.02').hypernyms()[0]
top = wn.synset('change.v.02')
while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
Synset('increase.v.01')
Synset('change_magnitude.v.01')
Synset('change.v.02')
```

Unlike nouns which all share a root hypernym, entity, root hypernyms for verbs are highly variable depending on the word. Otherwise, they are quite similar.

```
wn.morphy('grow')
```

```
'grow'
```

Wu-Palmer Similarity Metric

```
wn.synsets('cherry')
```

```
[Synset('cherry.n.01'),
 Synset('cherry.n.02'),
 Synset('cherry.n.03'),
 Synset('cerise.n.01'),
 Synset('red.s.01')]
```

```
wn.synsets('strawberry')
```

```
[Synset('strawberry.n.01'),
 Synset('strawberry.n.02'),
 Synset('strawberry.n.03')]
```

```
cherry = wn.synset('cherry.n.01')
strawberry = wn.synset('strawberry.n.01')
```

```
wn.wup_similarity(cherry, strawberry)
wn.wup_similarity(cherry, strawberry)

     0.35294117647058826
```

Lesk Algorithm

```
from nltk.wsd import lesk
```

```
for ss in wn.synsets('cherry'):
    print(ss, ss.definition())

     Synset('cherry.n.01') wood of any of various cherry trees especially the black cherry
     Synset('cherry.n.02') any of numerous trees and shrubs producing a small fleshy round fruit with a single hard stone; many also produce
     Synset('cherry.n.03') a red fruit with a single hard stone
     Synset('cerise.n.01') a red the color of ripe cherries
     Synset('red.s.01') of a color at the end of the color spectrum (next to orange); resembling the color of blood or cherries or tomatoes c
```

```
sent = ['Turn', 'right', 'at', 'the', 'cherry', 'red', 'house', '.']
print(lesk(sent, 'cherry', 'n'))
print(lesk(sent, 'cherry'))

     Synset('cherry.n.01')
     Synset('red.s.01')
```

```
for ss in wn.synsets('strawberry'):
    print(ss, ss.definition())

     Synset('strawberry.n.01') sweet fleshy red fruit
     Synset('strawberry.n.02') any of various low perennial herbs with many runners and bearing white flowers followed by edible fruits havir
     Synset('strawberry.n.03') a soft red birthmark
```

```
sent = ['The', 'baby', 'came', 'out', 'with', 'a', 'strawberry', '.']
print(lesk(sent, 'strawberry', 'n'))
print(lesk(sent, 'strawberry'))

     Synset('strawberry.n.03')
     Synset('strawberry.n.03')
```

Cherries and strawberries are both red fruits in the family rosaceae. I was surprised to see that the Wu-Palmer similarity metric gave them such a low score compared to dogs and cats (score of 0.86) which are in completely different families. It was also interesting to see how the Lesk algorithm might be making decisions. In our cherry example, running the Lask algorithm with the expectation of a noun made the algorithm consider the house and what material it might be constructed with more than the fruit definition of cherry which also includes the word "red".

SentiWordNet

SentiWordNet is an extension of WordNet that assigns sentiment scores to text. Outside of the obvious sentiment analyzing purposes, it can also be used for similar opinion mining purposes.

```
import nltk
nltk.download("sentiwordnet")
from nltk.corpus import sentiwordnet as swn

     [nltk_data] Downloading package sentiwordnet to /root/nltk_data...
     [nltk_data]   Unzipping corpora/sentiwordnet.zip.
```

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

     <ecstasy.n.01: PosScore=0.25 NegScore=0.125>
     <ecstasy.n.02: PosScore=0.125 NegScore=0.125>
     <adam.n.03: PosScore=0.0 NegScore=0.0>
```

```
for ss in wn.synsets('ecstasy'):
    print(ss, ss.definition())

     Synset('ecstasy.n.01') a state of being carried away by overwhelming emotion; - Charles Dickens
     Synset('ecstasy.n.02') a state of elated bliss
     Synset('adam.n.03') street names for methylenedioxymethamphetamine
```

```
sent = 'the worms on the wet pavement were writhing in ecstasy'
neg = 0
pos = 0
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        neg += syn.neg_score()
        pos += syn.pos_score()

print("neg\tpos counts")
print(neg, '\t', pos)

     neg     pos counts
     0.25     0.5
```

It was surprising to me how negative the word ecstasy scored. The first definition, to me, was extreme in a way that is balanced in both positive and negative connotations. The second definition includes the word "bliss" which seems very positive, yet SentiWordNet scored them as equal instead of the previous definition which was more negative. In the context of my sentence, ecstasy is more positive, like I would have expected before seeing the scores of the word alone. Knowing these scores would be extremely helpful considering how many applications of NLP involve sentiments and context in understanding human language.

Collocations

Collocations are words that have a different meaning when put together. For example, "popcorn shrimp" is a product of a specific method of preparing shrimp. You cannot replace the words with their synonyms and expect them to mean the same thing.

```
import nltk
nltk.download('webtext')
nltk.download('nps_chat')
from nltk.book import *
text4

     [nltk_data] Downloading package webtext to /root/nltk_data...
     [nltk_data]   Unzipping corpora/webtext.zip.
     [nltk_data] Downloading package nps_chat to /root/nltk_data...
     [nltk_data]   Package nps_chat is already up-to-date!
     *** 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
     <Text: Inaugural Address Corpus>
```

```
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]

     'Fellow - Citizens of the Senate and of the House o'
```

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

```
p(Old World) =  0.000997506234413965
p(Old) =  0.0010972568578553616
p(World) =  0.0017955112219451373
pmi =  8.983886091037398
```

The mutual information formula scored "Old World" fairly high which means that it is very likely to be a collocation. I agree with this result, as the Old World is a term that I'd heard and used a lot in history classes as a specific description of the past.

✓ 0s completed at 3:13 PM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.