WordNet

WordNet is a lexical database of nouns, verbs, adjectives, and adverbs that groups word into synonym sets called synsets. WordNet helps illustrate that people organize concepts and words into hierarchies.

Fruit

```
import nltk
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
True
```

```
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
True
```

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

Synsets for Fruit

```
wn.synsets('fruit')
```

```
[Synset('fruit.n.01'),
 Synset('yield.n.03'),
 Synset('fruit.n.03'),
 Synset('fruit.v.01'),
 Synset('fruit.v.02')]
```

fruit.n.03 definition() method

```
wn.synset('fruit.n.03').definition()
```

```
'the consequence of some effort or action'
```

examples() method

```
wn.synset('fruit.n.03').examples()
```

```
['he lived long enough to see the fruit of his policies']
```

lemmas() method

```
wn.synset('fruit.n.03').lemmas()
```

```
[Lemma('fruit.n.03.fruit')]
```

Traverse up the word hierarchy

```
fruit = wn.synset('fruit.n.03')
hyp = fruit.hypernyms()[0]
top = wn.synset('entity.n.01')
while hyp:
  print(hyp)
  if hyp == top:
    break
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
Synset('consequence.n.02')
Synset('result.n.03')
Synset('ending.n.04')
Synset('happening.n.01')
Synset('event.n.01')
Synset('psychological_feature.n.01')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

In WordNet, nouns are the most highly connected synsets. Nouns can be connected to other synsets with a hyponym-hypernym relationships which can branch off to also include meronym-holonym relationships.

```
wn.synset('fruit.n.03').hypernyms()
```

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

```
wn.synset('fruit.n.03').hyponyms()
```

```
[]
```

```
wn.synset('fruit.n.03').part_meronyms()
```

```
[]
```

```
wn.synset('fruit.n.03').part_holonyms()
```

```
[]
```

```
fruit.lemmas()[0].antonyms()
```

```
[]
```

Reading

Synsets for Reading

```
wn.synsets('reading')
```

```
[Synset('reading.n.01'),
 Synset('reading.n.02'),
 Synset('reading.n.03'),
 Synset('reading.n.04'),
 Synset('interpretation.n.01'),
 Synset('reading.n.06'),
 Synset('recitation.n.02'),
 Synset('reading.n.08'),
 Synset('read.v.01'),
 Synset('read.v.02'),
 Synset('read.v.03'),
 Synset('read.v.04'),
 Synset('read.v.05'),
 Synset('take.v.06'),
 Synset('learn.v.04'),
 Synset('read.v.08'),
 Synset('read.v.09'),
 Synset('read.v.10'),
 Synset('understand.v.03')]
```

read.v.01 definition() method

```
wn.synset('read.v.01').definition()
```

```
'interpret something that is written or printed'
```

examples() method

```
wn.synset('read.v.01').examples()
```

```
['read the advertisement', 'Have you read Salman Rushdie?']
```

lemmas() method

```
wn.synset('read.v.01').lemmas()
```

```
[Lemma('read.v.01.read')]
```

Traversing the hierarchy

```
read = wn.synset('read.v.01')
hyper = lambda s: s.hypernyms()
list(read.closure(hyper))
```

```
[Synset('interpret.v.01'), Synset('understand.v.01')]
```

In WordNet, verbs have less extensive synset relations than nouns. Verbs usually only have hypernym-hyponym relationships.

Using Morphy to find different forms of Read

```
wn.morphy('reading', wn.VERB)
```

```
'read'
```

```
wn.morphy('reading', wn.NOUN)
```

```
'reading'
```

Comparing Similar Words

Spoon and Fork

```
wn.synsets('spoon')
```

```
[Synset('spoon.n.01'),
 Synset('spoon.n.02'),
 Synset('spoon.n.03'),
 Synset('spoon.v.01'),
 Synset('smooch.v.01')]
```

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

```
'a piece of cutlery with a shallow bowl-shaped container and a handle; used to stir or serve or take up food'
```

```
wn.synsets('fork')
```

```
[Synset('fork.n.01'),
 Synset('branching.n.01'),
 Synset('fork.n.03'),
 Synset('fork.n.04'),
 Synset('crotch.n.02'),
 Synset('pitchfork.v.01'),
 Synset('fork.v.02'),
 Synset('branch.v.02'),
 Synset('fork.v.04')]
```

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

```
'cutlery used for serving and eating food'
```

Wu-Palmer similarity metric of spoon and fork

```
spoon = wn.synset('spoon.n.01')
fork = wn.synset('fork.n.01')
wn.wup_similarity(spoon, fork)
```

```
0.9
```

Lesk Algorithm

```
from nltk.wsd import lesk
sent = ['I', 'ate', 'my', 'soup', 'with', 'a', 'spoon', '.']
print(lesk(sent, 'spoon', 'n'))
```

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

```
wn.synset('spoon.n.03').definition()
```

```
'formerly a golfing wood with an elevated face'
```

```
sent = ['I', 'need', 'a', 'fork', 'to', 'eat', '.']
print(lesk(sent, 'fork', 'n'))
```

```
Synset('fork.n.04')
```

```
wn.synset('fork.n.04').definition()
```

```
'an agricultural tool used for lifting or digging; has a handle and metal prongs'
```

In this scenario, comparing spoon and fork, the Wu-Palmeralgorithm seemed to work best to find context and similarity between the nouns. The lesk algorithm provided incorrect synsets for the context of the sentence written, even though the sentences used words present in each of the word's definitions.

SentiWordNet is a lexical resource built on top of WordNet which analyzes the sentiments of a word in terms of positive, negative, and objective. This can be used to analyze texts, such as books or social media posts, and find out the tone and purpose of them.

```
import nltk
nltk.download('sentiwordnet')
```

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

```
from nltk.corpus import sentiwordnet as swn
rage_synsets = wn.synsets('rage')
print(rage_synsets)
```

```
3'), Synset('rage.n.04'), Synset('fad.n.01'), Synset('ramp.v.01'), Synset('rage.v.02'), S
```

```
wn.synset('rage.n.02').definition()
```

```
'a state of extreme anger'
```

```
print(swn.senti_synset('fury.n.01'))
```

```
<fury.n.01: PosScore=0.25 NegScore=0.5>
```

```
print(swn.senti_synset('rage.n.02'))
```

```
<rage.n.02: PosScore=0.0 NegScore=0.125>
```

```
print(swn.senti_synset('rage.n.03'))
```

```
<rage.n.03: PosScore=0.625 NegScore=0.0>
```

```
print(swn.senti_synset('rage.n.04'))
```

```
      <rage.n.04: PosScore=0.0 NegScore=0.125>

print(swn.senti_synset('fad.n.01'))

      <fad.n.01: PosScore=0.25 NegScore=0.0>


print(swn.senti_synset('ramp.v.01'))

      <ramp.v.01: PosScore=0.0 NegScore=0.0>


print(swn.senti_synset('ramp.v.02'))

      <ramp.v.02: PosScore=0.0 NegScore=0.0>


print(swn.senti_synset('rage.v.03'))

      <rage.v.03: PosScore=0.0 NegScore=0.5>


sent = 'that was such a fun roller coaster'
neg = 0
pos = 0
tokens = sent.split()
for token in tokens:
  print(token)
  syn_list = list(swn.senti_synsets(token))
  if syn_list:
    syn = syn_list[0]
    print(syn)


     that
     was
     <washington.n.02: PosScore=0.0 NegScore=0.0>
     such
     <such.s.01: PosScore=0.0 NegScore=0.125>
     a
     <angstrom.n.01: PosScore=0.0 NegScore=0.0>
     fun
     <fun.n.01: PosScore=0.375 NegScore=0.0>
     roller
     <roller.n.01: PosScore=0.0 NegScore=0.0>
     coaster
     <coaster.n.01: PosScore=0.0 NegScore=0.0>
```

The scores show that in most sentences, there are only a couple of words that demonstrate the overall sentiment of a sentence. In this example only 'such' and 'fun' contributed to the sentiment score of the sentence. Knowing these scores will help understand the types of words to look for in larger databases.

Collocation

A collocation is when two or more words appear next to each other, more than would be expected by each word's basic definition, to form a complete phrase with its own meaning. The words used in a collocation cannot be subsituted with a synonym of those words because that would not result in the same phrase meaning.

```
nltk.download('book')
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.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 ieer to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ieer.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 nps_chat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
```

```
import nltk.book
```

```
*** 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
```

```
from nltk.book import *
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 for collocation: **fellow citizens**

log2 [P(x, y)]/[P(x)*P(y)] = log2 (61/149796)/(128/149797)*(240/149797) = 8.2

A larger positive number indicates that "fellow citizens" is a collocation because it occurs together more often than expected and many times throughout the Text.

Colab paid products - Cancel contracts here

✓ 0s completed at 8:47 PM