```
import nltk
```

Wordnet is a database of nouns, adjectives, verbs, and adverbs that provides glosses and uses examples. It groups words into synonym sets that it calls synsets.

```
nltk.download("stopwords")
nltk.download("wordnet")
nltk.download("punkt")

nltk.download("omw-1.4")
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
from nltk.book import *
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt 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 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!
```

```
wordnet.synsets('chair')
```

```
[Synset('chair.n.01'),
 Synset('professorship.n.01'),
 Synset('president.n.04'),
 Synset('electric_chair.n.01'),
 Synset('chair.n.05'),
 Synset('chair.v.01'),
 Synset('moderate.v.01')]
```

```
wordnet.synset('chair.n.01').definition()
```

```
'a seat for one person, with a support for the back'
```

```
wordnet.synset('chair.n.01').examples()
```

```
['he put his coat over the back of the chair and sat down']
```

```
wordnet.synset('chair.n.01').lemmas()
```

```
[Lemma('chair.n.01.chair')]
```

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

```
Synset('chair.n.01')
Synset('seat.n.03')
Synset('furniture.n.01')
Synset('furnishing.n.02')
Synset('instrumentality.n.03')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Wordnet organizes nouns in a hierarchial manner. The synset gets more and more broader until it reaches "entity".

```
wordnet.synset('chair.n.01').hypernyms()
```

```
[Synset('seat.n.03')]
```

```
wordnet.synset('chair.n.01').hyponyms()
```

```
[Synset('armchair.n.01'),
 Synset('barber_chair.n.01'),
 Synset('chair_of_state.n.01'),
 Synset('chaise_longue.n.01'),
 Synset('eames_chair.n.01'),
 Synset('fighting_chair.n.01'),
 Synset('folding_chair.n.01'),
```

```
      Synset('highchair.n.01'),
      Synset('ladder-back.n.01'),
      Synset('lawn_chair.n.01'),
      Synset('rocking_chair.n.01'),
      Synset('straight_chair.n.01'),
      Synset('swivel_chair.n.01'),
      Synset('tablet-armed_chair.n.01'),
      Synset('wheelchair.n.01')]


wordnet.synset('chair.n.01').part_meronyms()

     [Synset('back.n.08'), Synset('leg.n.03')]


wordnet.synset('chair.n.01').substance_meronyms()

     []


wordnet.synset('chair.n.01').part_holonyms()

     []


wordnet.synset('chair.n.01').substance_holonyms()

     []


synset = wordnet.synset("chair.n.01")
for i in synset.lemmas():

    print(i.antonyms())

     []


wordnet.synsets('run')

     [Synset('run.n.01'),
      Synset('test.n.05'),
      Synset('footrace.n.01'),
      Synset('streak.n.01'),
      Synset('run.n.05'),
      Synset('run.n.06'),
      Synset('run.n.07'),
      Synset('run.n.08'),
      Synset('run.n.09'),
      Synset('run.n.10'),
      Synset('rivulet.n.01'),
      Synset('political_campaign.n.01'),
      Synset('run.n.13'),
      Synset('discharge.n.06'),
      Synset('run.n.15'),
      Synset('run.n.16'),
```

```
 Synset('run.v.01'),
 Synset('scat.v.01'),
 Synset('run.v.03'),
 Synset('operate.v.01'),
 Synset('run.v.05'),
 Synset('run.v.06'),
 Synset('function.v.01'),
 Synset('range.v.01'),
 Synset('campaign.v.01'),
 Synset('play.v.18'),
 Synset('run.v.11'),
 Synset('tend.v.01'),
 Synset('run.v.13'),
 Synset('run.v.14'),
 Synset('run.v.15'),
 Synset('run.v.16'),
 Synset('prevail.v.03'),
 Synset('run.v.18'),
 Synset('run.v.19'),
 Synset('carry.v.15'),
 Synset('run.v.21'),
 Synset('guide.v.05'),
 Synset('run.v.23'),
 Synset('run.v.24'),
 Synset('run.v.25'),
 Synset('run.v.26'),
 Synset('run.v.27'),
 Synset('run.v.28'),
 Synset('run.v.29'),
 Synset('run.v.30'),
 Synset('run.v.31'),
 Synset('run.v.32'),
 Synset('run.v.33'),
 Synset('run.v.34'),
 Synset('ply.v.03'),
 Synset('hunt.v.01'),
 Synset('race.v.02'),
 Synset('move.v.13'),
 Synset('melt.v.01'),
 Synset('ladder.v.01'),
 Synset('run.v.41')]
```

```
wordnet.synset('run.v.01').definition()
```

```
'move fast by using one's feet, with one foot off the ground at any given time'
```

```
wordnet.synset('run.v.01').examples()
```

```
["Don't run--you'll be out of breath", 'The children ran to the store']
```

```
wordnet.synset('run.v.01').lemmas()
```

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

```
wordnet.synset('run.v.01').hypernyms()
```

```
[Synset('travel_rapidly.v.01')]
```

```
wordnet.synset('travel_rapidly.v.01').hypernyms()
```

```
[Synset('travel.v.01')]
```

```
wordnet.synset('travel.v.01').hypernyms()
```

```
[]
```

Verbs do not have a top-level entity like nouns. Traversing requires going through the hypernyms one by one and seeing where the chain ends.

```
wordnet.morphy('run', wordnet.VERB)
```

```
'run'
```

```
wordnet.morphy('run', wordnet.NOUN)
```

```
'run'
```

```
wordnet.morphy('running', wordnet.VERB)
```

```
'run'
```

```
wordnet.morphy('run', wordnet.ADJ)
```

```
wordnet.synsets('run')
```

```
[Synset('run.n.01'),
 Synset('test.n.05'),
 Synset('footrace.n.01'),
 Synset('streak.n.01'),
 Synset('run.n.05'),
 Synset('run.n.06'),
 Synset('run.n.07'),
 Synset('run.n.08'),
 Synset('run.n.09'),
 Synset('run.n.10'),
 Synset('rivulet.n.01'),
 Synset('political_campaign.n.01'),
 Synset('run.n.13'),
```

```
 Synset('discharge.n.06'),
 Synset('run.n.15'),
 Synset('run.n.16'),
 Synset('run.v.01'),
 Synset('scat.v.01'),
 Synset('run.v.03'),
 Synset('operate.v.01'),
 Synset('run.v.05'),
 Synset('run.v.06'),
 Synset('function.v.01'),
 Synset('range.v.01'),
 Synset('campaign.v.01'),
 Synset('play.v.18'),
 Synset('run.v.11'),
 Synset('tend.v.01'),
 Synset('run.v.13'),
 Synset('run.v.14'),
 Synset('run.v.15'),
 Synset('run.v.16'),
 Synset('prevail.v.03'),
 Synset('run.v.18'),
 Synset('run.v.19'),
 Synset('carry.v.15'),
 Synset('run.v.21'),
 Synset('guide.v.05'),
 Synset('run.v.23'),
 Synset('run.v.24'),
 Synset('run.v.25'),
 Synset('run.v.26'),
 Synset('run.v.27'),
 Synset('run.v.28'),
 Synset('run.v.29'),
 Synset('run.v.30'),
 Synset('run.v.31'),
 Synset('run.v.32'),
 Synset('run.v.33'),
 Synset('run.v.34'),
 Synset('ply.v.03'),
 Synset('hunt.v.01'),
 Synset('race.v.02'),
 Synset('move.v.13'),
 Synset('melt.v.01'),
 Synset('ladder.v.01'),
 Synset('run.v.41')]
```

```
wordnet.synsets('dash')
```

```
[Synset('dash.n.01'),
 Synset('dash.n.02'),
 Synset('dash.n.03'),
 Synset('hyphen.n.01'),
 Synset('dash.n.05'),
 Synset('dash.n.06'),
 Synset('dart.v.02'),
 Synset('smash.v.02'),
```

```
       Synset('crash.v.10'),
       Synset('dash.v.04'),
       Synset('daunt.v.01'),
       Synset('dash.v.06')]


run = wordnet.synset('run.v.23')
dash = wordnet.synset('dash.v.06')

wordnet.wup_similarity(run, dash)
```

```
     0.2
```

```
from nltk.wsd import lesk
sentence = "I run to the bank"
print(lesk(sentence, 'run', 'v'))
sentence = "I dash to the bank"
print(lesk(sentence, 'dash', 'v'))
```

```
     Synset('tend.v.01')
     Synset('smash.v.02')
```

My observations are that the Wu Palmer algorithm did not find much similarity between the two words so I might have chosen a run with a different definition than the one I intended. The lesk algorithm suggested tend and smash however those words do not mean run and dash respectively so the algorithm is not able to find the correct meaning for the ambiguous word for some reason.

Senti wordnet can be used to assign positivity, negativity, and objectivity scores to a synset. It can be used to iterate over tokens and see how many have positive sentiment and negative sentiment.

```
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
nltk.download('wordnet')
nltk.download('omw-1.4')
cry = swn.senti_synset('cry.v.03')
```

```
     [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...
```

```
print(cry)
print("Positive score = ", cry.pos_score())
print("Negative score = ", cry.neg_score())
print("Objective score = ", cry.obj_score())
```

```
<exclaim.v.01: PosScore=0.125 NegScore=0.5>
Positive score =  0.125
Negative score =  0.5
Objective score =  0.375
```

```
sent = 'I am crying today and happy tomorrow'
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
0.0      1.0
```

My observations of these scores is that the senti wordnet viewed cry as similar to exclaim and says that exclaim has a higher negative than positive score. In the sentence, it said the sentence was postive scored even with the word crying. The utility of knowing these scores is that you can scan a passage and understand whether the author is writing with a positive or negative sentiment which can help give you data about the author.

A collocation is when two words are found frequently together. A key to finding collocations is that the two words cannot be swapped for synonyms.

```
nltk.download('genesis')
nltk.download('gutenberg')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('stopwords')

from nltk.book import *
text4.collocations()
```

```
text4.collocations()
```

```
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Package genesis 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 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]   Unzipping corpora/stopwords.zip.
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
```

Picking the phrase fellow citizens, the mutual information gives a pmi score of 8.2. I think that this means that fellow citizens is a collocation since the score is positive. The formula uses independence in probability and suggests that a pmi value of zero means that x and y are independent and therefore are not a collocation. A pmi value that is positive means that x and y are together more than expected and therefore is a collocation, and a pmi value that is negative means that x and y are not a collocation

✓ 0s completed at 6:57 PM