

```
from nltk.corpus import wordnet as wn
import nltk
nltk.download('wordnet')
nltk.download('all')
```

WordNet is an organization of words based on category and related by their meanings. Words each have short definitions and can be related to other words using synonym sets.

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

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

```
chosen_synset = noun_synsets[0]
print("Definition: " + str(chosen_synset.definition()))
print("Examples: " + str(chosen_synset.examples()))
print("Lemmas: " + str(chosen_synset.lemmas()))
```

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

```
hyper = lambda s: s.hypernyms()
list(chosen_synset.closure(hyper))
```

```
[Synset('canine.n.02'),
 Synset('domestic_animal.n.01'),
 Synset('carnivore.n.01'),
 Synset('animal.n.01'),
 Synset('placental.n.01'),
 Synset('organism.n.01'),
 Synset('mammal.n.01'),
 Synset('living_thing.n.01'),
 Synset('vertebrate.n.01'),
 Synset('whole.n.02'),
 Synset('chordate.n.01'),
 Synset('object.n.01'),
 Synset('physical_entity.n.01'),
 Synset('entity.n.01')]
```

For nouns, every noun derives from the synset 'entity.n.01' which is at the top of the hierarchy. Entity splits into a few categories, like physical_entity and thing, and each splits up from there to form the entire corpus of nouns. However, nouns can have multiple parent nouns, which is why the ordering seems odd above.

✓ 0s completed at 9:32 PM



```
print("Hypernyms: " + str(chosen_synset.hypernyms()))
print("Hyponyms: " + str(chosen_synset.hyponyms()))
print("Meronyms: " + str(chosen_synset.part_meronyms() + chosen_synset.substance_meronyms()))
print("Holonyms: " + str(chosen_synset.part_holonyms() + chosen_synset.substance_holonyms()))
print("Antonyms: " + str(chosen_synset.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'), Synset('
Meronyms: [Synset('flag.n.07')]
Holonyms: []
Antonyms: []
```

```
verb = "throw"
verb_synsets = wn.synsets(verb, pos=wn.VERB)
print(verb_synsets)
```

```
[Synset('throw.v.01'), Synset('throw.v.02'), Synset('shed.v.01'), Synset('throw.v.04'
```

```
chosen_synset = verb_synsets[0]
print("Definition: " + str(chosen_synset.definition()))
print("Examples: " + str(chosen_synset.examples()))
print("Lemmas: " + str(chosen_synset.lemmas()))
```

```
Definition: propel through the air
Examples: ['throw a frisbee']
Lemmas: [Lemma('throw.v.01.throw')]
```

```
hyper = lambda s: s.hypernyms()
list(chosen_synset.closure(hyper))
```

```
[Synset('propel.v.01'), Synset('move.v.02')]
```

While nouns have one hierarchy with one parent ("entity.n.01"), verbs have no such single hierarchy. Verbs are separated into multiple hierarchies; in this case the highest verb is "move".

```
print(wn.morphy("throw", wn.VERB))
print(wn.morphy("throwing", wn.VERB))
print(wn.morphy("thrown", wn.VERB))
print(wn.morphy("threw", wn.VERB))
```

```
throw
throw
throw
throw
```

```
similar_word_1 = wn.synset("toss.v.01")
```

```

similar_word_2 = wn.synset("drop.v.01")
print("Wu-Palmer similarity 1: %.2f" % chosen_synset.wup_similarity(similar_word_1))
print("Wu-Palmer similarity 2: %.2f" % chosen_synset.wup_similarity(similar_word_2))
from nltk.wsd import lesk
print(lesk("I am going to throw this ball.", "throw"), lesk("I am going to throw this ball.", "toss"), lesk("I am going to toss this ball.", "drop"), lesk("I am going to drop this ball.", "throw"))

Wu-Palmer similarity 1: 0.75
Wu-Palmer similarity 2: 0.29
Synset('throw.v.14') throw (a die) out onto a flat surface
Synset('flip.v.06') throw or toss with a light motion
Synset('sink.v.01') fall or descend to a lower place or level

```

The Wu-Palmer similarities are as expected. I know "toss" is more similar to "throw" than "drop" is, so the number disparity is expected. However, lesk seemed to have performed at a mediocre level and using interesting choices of synsets. It is guessing that I mean to throw a ball as if it were a die, for example.

SentiWordNet analyzes the sentiment of words--positive, negative, or neutral. This sentiment analysis is integral in understanding human language, as it helps convey meaning if a computer understands overall opinion from a sentence instead of just parsing it as facts.

```

from nltk.corpus import sentiwordnet as swn
emotionally_charged_word = "hate"
for senti_synset in swn.senti_synsets(emotionally_charged_word):
    print(senti_synset)

<hate.n.01: PosScore=0.125 NegScore=0.375>
<hate.v.01: PosScore=0.0 NegScore=0.75>

sentence = "I am going to go crazy if I see that dastardly dog on my porch again."
for word in nltk.word_tokenize(sentence):
    senti_synset_list = list(swn.senti_synsets(word))
    for senti_synset in senti_synset_list:
        print(senti_synset, end=" ")
    print()

<iodine.n.01: PosScore=0.0 NegScore=0.0><one.n.01: PosScore=0.0 NegScore=0.0><i.n.03: PosScore=0.0 NegScore=0.0><americium.n.01: PosScore=0.0 NegScore=0.0><master_of_arts.n.01: PosScore=0.0 NegScore=0.0><departure.n.01: PosScore=0.0 NegScore=0.0><passing.n.02: PosScore=0.0 NegScore=0.25>

<go.n.01: PosScore=0.0 NegScore=0.0><adam.n.03: PosScore=0.0 NegScore=0.0><crack.n.09: PosScore=0.0 NegScore=0.0><crazy.n.01: PosScore=0.0 NegScore=0.375><brainsick.s.01: PosScore=0.0 NegScore=0.5><

<iodine.n.01: PosScore=0.0 NegScore=0.0><one.n.01: PosScore=0.0 NegScore=0.0><i.n.03: PosScore=0.0 NegScore=0.0><see.n.01: PosScore=0.0 NegScore=0.0><see.v.01: PosScore=0.0 NegScore=0.0><understand.n.01: PosScore=0.0 NegScore=0.0>

```

```

<dastard.s.01: PosScore=0.125 NegScore=0.5>
<dog.n.01: PosScore=0.0 NegScore=0.0><frump.n.01: PosScore=0.0 NegScore=0.5><dog.n.03
<on.a.01: PosScore=0.0 NegScore=0.0><on.a.02: PosScore=0.0 NegScore=0.0><along.r.01:

<porch.n.01: PosScore=0.0 NegScore=0.0>
<again.r.01: PosScore=0.0 NegScore=0.0>

```

Naively printing from the first synset in the index gives bad results because they are rarely correct. Instead, I printed all the synsets from any given word in the sentence. It seems to have parsed sentiment accurately given the right definition is chosen, e.g. "dastardly" is 50% negative and only 12.5% positive. Nevertheless, if the right definition is chosen, sentiment analysis has many uses in NLP, such as customer support being better tuned to understand customer emotion, or other human-interfacing systems having the same capability, such as virtual assistants.

A collocation is a combination of two or more words which occurs more often than they would by random chance, signifying that they have some meaning when used together. With collocations, substitution of one word for a synonym often does not work properly.

```

from nltk.book import text4
print(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
None


```

```

selected_collocation = "fellow citizens"
import math
def calculate_pmi(bigram, text):
    word1, word2 = bigram.split()
    bigrams = nltk.bigrams(text)
    frequency = nltk.FreqDist(bigrams)
    numerator = frequency[(word1, word2)] / (len(text4) - 1)
    denominator = (text.count(word1) / len(text)) * (text.count(word2) / len(text))
    quotient = numerator / denominator
    return math.log(quotient, 2)

print(calculate_pmi(selected_collocation, text4))
print(calculate_pmi("is the", text4))

```

 8.174006563041724

0.9795481237356592

The mutual information formula as I implemented it gave a positive PMI of 8.17, which is to be expected because "fellow citizens" has a meaning that is greater than the sum of its parts. When I try the same calculation with something like "is the", I don't get as strong of a positive result because those words don't have special meaning when put together.

[Colab paid products](#) - [Cancel contracts here](#)