What is WordNet?

WordMet is a database which organizes English word and their definitions into groups of like-meaning words (synsets) and example usages. It assists in natural language processing tasks such as translation.

```
import nltk
#nltk.download('wordnet')
#nltk.download('omw-1.4')
from nltk.corpus import wordnet

#selecting random noun
synsets = wordnet.synsets('monitor', pos = 'n')

for synset in synsets:
    print(synset)
```

```
Synset('proctor.n.01')
Synset('admonisher.n.01')
Synset('monitor.n.03')
Synset('monitor.n.04')
Synset('monitor.n.05')
Synset('monitor.n.06')
Synset('monitor.n.07')
```

```
selected_synset = synsets[0]

print("The definition is:", selected_synset.definition())
print("The examples are:", selected_synset.examples())
print("The lemmas are:", selected_synset.lemma_names(), "\n")

#traversing up the WordNet hierarchy
hypernyms = selected_synset.hypernyms()
print("The synsets are: ")
while len(hypernyms) != 0:
    print(hypernyms)
    hypernyms = hypernyms[0].hypernyms()
```

```
The definition is: someone who supervises (an examination)
The examples are: []
The lemmas are: ['proctor', 'monitor']

The synsets are:
[Synset('supervisor.n.01')]
[Synset('superior.n.01')]
[Synset('leader.n.01')]
[Synset('person.n.01')]
[Synset('causal_agent.n.01'), Synset('organism.n.01')]
```

✓ 0s completed at 5:39 PM ● ×

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

Observations of how WordNet is setup for nouns:

- The nouns are setup in a hieracrhy where, as you go up, the nouns get more general/broad
- The hierarchy helps group nouns based on their meanings

```
#Get the hypernyms
hypernyms = selected_synset.hypernyms()
print("Hypernyms: ", end="")
if len(hypernyms) != 0:
    print(hypernyms)

# Get the hyponyms
hyponyms = selected_synset.hyponyms()
print("Hyponyms: ", end="")
if len(hyponyms) != 0:
    print(hyponyms)

# Get the meronyms
meronyms = selected_synset.part_meronyms()
print("Meronyms: ", end="")
if len(meronyms) != 0:
    print(meronyms)
else:
    print("")

# Get the holonyms
hyponyms = selected_synset.hyponyms()
print("Hyponyms: ", end="")
if len(hyponyms) != 0:
    print(hyponyms)

# Get the antonyms
lemmas = selected_synset.lemmas()
print("Antonyms: ", end="")
if len(lemmas) != 0:
  for lemma in lemmas:
    if len(lemma.antonyms())!=0:
      print(lemma.antonyms())
```

```
Hypernyms: [Synset('supervisor.n.01')]
Hyponyms: [Synset('invigilator.n.01')]
Meronyms:
Hyponyms: [Synset('invigilator.n.01')]
Antonyms:
```

```
#selecting random verb
synsets = wordnet.synsets('see', pos = 'v')

for synset in synsets:
    print(synset)
```

```
Synset('see.v.01')
Synset('understand.v.02')
Synset('witness.v.02')
Synset('visualize.v.01')
Synset('see.v.05')
Synset('learn.v.02')
Synset('watch.v.03')
Synset('meet.v.01')
Synset('determine.v.08')
Synset('see.v.10')
Synset('see.v.11')
Synset('see.v.12')
Synset('visit.v.01')
Synset('attend.v.02')
Synset('see.v.15')
Synset('go_steady.v.01')
Synset('see.v.17')
Synset('see.v.18')
Synset('see.v.19')
Synset('examine.v.02')
Synset('experience.v.01')
Synset('see.v.22')
Synset('see.v.23')
Synset('interpret.v.01')
```

```
selected_synset = synsets[0]

print("The definition is:", selected_synset.definition())
print("The examples are:", selected_synset.examples())
print("The lemmas are:", selected_synset.lemma_names(), "\n")

#traversing up the WordNet hierarchy
hypernyms = selected_synset.hypernyms()
print("The synsets are: ")
while len(hypernyms) != 0:
    print(hypernyms)
    hypernyms = hypernyms[0].hypernyms()
```

```
The definition is: perceive by sight or have the power to perceive by sight
The examples are: ['You have to be a good observer to see all the details', 'Can you
The lemmas are: ['see']

The synsets are:
[Synset('perceive.v.01')]
```

Observations of how WordNet is setup for verbs:

Observations of how WordNet is setup for verbs:

- Similar to how the noun hierachy is setup (but perhpas with less depth in the hierarchy)
    - The verbs are setup in a hieracrhy where, as you go up, the verbs get more general/broad
    - The hierarchy helps group verbs based on their meanings

```
synsets
```

```
[Synset('see.v.01'),
 Synset('understand.v.02'),
 Synset('witness.v.02'),
 Synset('visualize.v.01'),
 Synset('see.v.05'),
 Synset('learn.v.02'),
 Synset('watch.v.03'),
 Synset('meet.v.01'),
 Synset('determine.v.08'),
 Synset('see.v.10'),
 Synset('see.v.11'),
 Synset('see.v.12'),
 Synset('visit.v.01'),
 Synset('attend.v.02'),
 Synset('see.v.15'),
 Synset('go_steady.v.01'),
 Synset('see.v.17'),
 Synset('see.v.18'),
 Synset('see.v.19'),
 Synset('examine.v.02'),
 Synset('experience.v.01'),
 Synset('see.v.22'),
 Synset('see.v.23'),
 Synset('interpret.v.01')]
```

```
different_forms = []

for synset in synsets:

  for lemma in synset.lemmas():
    if lemma.name() not in different_forms:
      different_forms.append(lemma.name())

      for related_lemma in lemma.derivationally_related_forms():
        if related_lemma.name() not in different_forms:
          different_forms.append(related_lemma.name())

print("Related words:")
for word in different_forms:
  print(word)
```

```
Related words:
```

```
Related words:
see
seer
seeing
understand
understandable
understanding
realize
realization
realise
realisation
witness
find
finder
visualize
visualization
visualizer
visualise
envision
envisioning
project
fancy
figure
figuration
picture
picturing
image
imaging
imagery
consider
reckon
view
regard
learn
hear
get_word
get_wind
pick_up
find_out
get_a_line
discover
discovery
watch
catch
take_in
meet
meeting
run_into
encounter
run_across
come_across
determine
determination
check
ascertain
ascertainable
```

```
     ascertainable
     insure
     see_to_it
```

```
word1 = "cat"
word2 = "kitten"

synsets1 = wordnet.synsets(word1, pos = 'n')
synsets2 = wordnet.synsets(word2, pos = 'n')

"""
for synset in synsets1:
    print(synset)

print()
for synset in synsets2:
    print(synset)
"""

selected_synset1 = synsets1[0]
selected_synset2 = synsets2[0]

print("selected synset 1", selected_synset1)
print("selected synset 2", selected_synset2)
```

```
     selected synset 1 Synset('cat.n.01')
     selected synset 2 Synset('kitten.n.01')
```

```
print("The Wu-Palmer similarity metric is", selected_synset1.wup_similarity(selected_synse
```

```
     The Wu-Palmer similarity metric is 0.5833333333333334
```

```
from nltk.wsd import lesk
sentence = "A cat likes playing and drinking milk"

synset = lesk(selected_synset1.name(), sentence)
print(synset)
```

```
     None
```

I thought it was interesting that the wu-palmer similarity metric wasn't higher. It still indicates similarity, however, which is expected

Double-click (or enter) to edit

```
from nltk.corpus import sentiwordnet
from nltk.tokenize import word_tokenize
#nltk.download('sentiwordnet')
```

```
#nltk.download('sentiwordnet')
#nltk.download('punkt')
hate_synsets = sentiwordnet.senti_synsets('hate')
for synset in hate_synsets:
  print( "The synset is", synset.synset.name(), ", the positive polarity score is", synset.
print()

example_sentence = "I love and hate the atomic composition of pickles."

for word in word_tokenize(example_sentence):
  synsets = sentiwordnet.senti_synsets(word)
  print("For", word, ", the positive polarity score is", synset.pos_score(), ", the negati
```

```
    The synset is hate.n.01 , the positive polarity score is 0.125 , the negative polarit
    The synset is hate.v.01 , the positive polarity score is 0.0 , the negative polarity 

    For I , the positive polarity score is 0.0 , the negative polarity score is 0.75 , th
    For love , the positive polarity score is 0.0 , the negative polarity score is 0.75 ,
    For and , the positive polarity score is 0.0 , the negative polarity score is 0.75 ,
    For hate , the positive polarity score is 0.0 , the negative polarity score is 0.75 ,
    For the , the positive polarity score is 0.0 , the negative polarity score is 0.75 ,
    For atomic , the positive polarity score is 0.0 , the negative polarity score is 0.75
    For composition , the positive polarity score is 0.0 , the negative polarity score is
    For of , the positive polarity score is 0.0 , the negative polarity score is 0.75 , t
    For pickles , the positive polarity score is 0.0 , the negative polarity score is 0.7
    For . , the positive polarity score is 0.0 , the negative polarity score is 0.75 , th
```

These scores would be very useful in sentiment analysis and categorizing the tone of a sentece. This can help score comments and validate the accuracy of a NLP model

A collocation is series of words that frequently appear together. They can provide context into understanding languages and help with NLP processing as there appearance usually has more reason for appearing than just probability.

```
#nltk.download('inaugural')
#nltk.download('stopwords')
#nltk.download('gutenberg')
#nltk.download('webtext')
##nltk.download('nps_chat')
#nltk.download('treebank')
#nltk.download('genesis')
from nltk.book import text4
from nltk.text import Text
from nltk.collocations import BigramAssocMeasures, BigramCollocationFinder

example_text = Text(text4)
collocations = example_text.collocation_list()
print(collocations)
print(collocations)

selected_collocation = collocations[1]
print("\n", selected_collocation)
```

```
[('United', 'States'), ('fellow', 'citizens'), ('years', 'ago'), ('four', 'years'), (

 ('fellow', 'citizens')
```