Wordnet Summary

Wordnet is a database and organizational system included with NLTK. It provides a hierarchical system of words, and helps with finding synonyms, antonyms, hypernyms, hyponyms, and more.

Synsets of a noun

In [1]:

```
from nltk.corpus import wordnet

# Get synsets of a noun
wordnet.synsets('dog')
```

Out[1]:

```
[Synset('dog.n.01'),
 Synset('frump.n.01'),
 Synset('dog.n.03'),
 Synset('cad.n.01'),
 Synset('frank.n.02'),
 Synset('pawl.n.01'),
 Synset('andiron.n.01'),
 Synset('chase.v.01')]
```

Definition, usage sample, and lemmas for dog.n.01

In [2]:

```
print("Definition:", wordnet.synset('dog.n.01').definition())
print("\nExample Usage:", wordnet.synset('dog.n.01').examples())
print("\nLemmas", wordnet.synset('dog.n.01').lemmas())
```

```
Definition: a member of the genus Canis (probably descended from the comm
on wolf) that has been domesticated by man since prehistoric times; occur
s in many breeds

Example Usage: ['the dog barked all night']

Lemmas [Lemma('dog.n.01.dog'), Lemma('dog.n.01.domestic_dog'), Lemma('do
g.n.01.Canis_familiaris')]
```

Hierarchy of synsets for dog.n.01

In [3]:

```
# Create hyper, and then call using closure
hyper = lambda s: s.hypernyms()
dog = wordnet.synset('dog.n.01')
list(dog.closure(hyper))
```

```
/Users/noahwhitworth/Library/Python/3.10/lib/python/site-packages/nltk/co
rpus/reader/wordnet.py:604: UserWarning: Discarded redundant search for S
ynset('animal.n.01') at depth 7
  for synset in acyclic_breadth_first(self, rel, depth):
```

Out[3]:

```
[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')]
```

Wordnet organizes its nouns in a hierarchical manner. That is, for each noun, there is a hypernym that the noun is an example of (for example, a dog is an animal, and animals are organisms, etc.).

Outputting hypernyms, hyponyms, meronyms, holonyms, and antonyms.

In [4]:
```
dog = wordnet.synset('dog.n.01')
print("Hypernyms:", dog.hypernyms())
print("Hyponyms:", dog.hyponyms())
print("Meronyms:", dog.part_meronyms())
print("Holonyms:", dog.part_holonyms())

# Get the lemma before finding the antonym
dog_lemma = wordnet.synset('dog.n.01').lemmas()
print("Antonyms:", dog_lemma[0].antonyms())
```

```
Hypernyms: [Synset('canine.n.02'), Synset('domestic_animal.n.01')]
Hyponyms: [Synset('basenji.n.01'), Synset('corgi.n.01'), Synset('cur.n.0
1'), Synset('dalmatian.n.02'), Synset('great_pyrenees.n.01'), Synset('gri
ffon.n.02'), Synset('hunting_dog.n.01'), Synset('lapdog.n.01'), Synset('l
eonberg.n.01'), Synset('mexican_hairless.n.01'), Synset('newfoundland.n.0
1'), Synset('pooch.n.01'), Synset('poodle.n.01'), Synset('pug.n.01'), Syn
set('puppy.n.01'), Synset('spitz.n.01'), Synset('toy_dog.n.01'), Synset
('working_dog.n.01')]
Meronyms: [Synset('flag.n.07')]
Holonyms: []
Antonyms: []
```

Synsets of a verb.

We will use "hop" as our verb.

In [5]:
```
wordnet.synsets('hop')
```

Out[5]:
```
[Synset('hop.n.01'),
 Synset('hop.n.02'),
 Synset('hop.n.03'),
 Synset('hop.v.01'),
 Synset('hop.v.02'),
 Synset('hop.v.03'),
 Synset('hop.v.04'),
 Synset('hop.v.05'),
 Synset('hop.v.06')]
```

Definition, usage sample, and lemmas for hop.v.05

In [6]:

```
print("Definition:", wordnet.synset('hop.v.05').definition())
print("\nExample Usage:", wordnet.synset('hop.v.05').examples())
print("\nLemmas", wordnet.synset('hop.v.05').lemmas())
```

```
Definition: jump across

Example Usage: ['He hopped the bush']

Lemmas [Lemma('hop.v.05.hop')]
```

Heirarchy for the verb

In [7]:

```
# Create hyper, and then call using closure
hyper = lambda s: s.hypernyms()
hop = wordnet.synset('hop.v.05')
list(hop.closure(hyper))
```

Out[7]:

```
[Synset('clear.v.09'),
 Synset('pass.v.07'),
 Synset('advance.v.01'),
 Synset('travel.v.01')]
```

Verbs are organized similarly to how nouns are organized, in that they get more and more broad as we move up the hierarchy. However, we can notice that there is no sysnet displayed here that would encompass all verbs. This contrasts with nouns, which all have "entity" as the top level synset.

Using Morphy to determine the different forms of the word

In [8]:

```
print(wordnet.morphy('hop', wordnet.VERB))
```

```
hop
```

Wu-Palmer and Lesk algorithms

In [9]:

```python
from nltk.wsd import lesk

hop = wordnet.synset('hop.v.01')
jump = wordnet.synset('jump.v.01')

# Wu-Palmer
print("Result of Wu-Palmer: ",wordnet.wup_similarity(hop, jump))

print("\nSynsets for hop:")
for i in wordnet.synsets('hop'):
    print(i, i.definition())

print("\nSynsets for jump:")
for i in wordnet.synsets('jump'):
    print(i, i.definition())

# Lesk
print("Sentence used for Lesk: The rabbit hopped into the air. \n")
print("\nResult of lesk:", lesk('The rabbit hopped into the air.', 'jump'))
print("\nDefinition:", wordnet.synset('startle.v.02').definition())
```

```
Result of Wu-Palmer:  0.8

Synsets for hop:
Synset('hop.n.01') the act of hopping; jumping upward or forward (especia
lly on one foot)
Synset('hop.n.02') twining perennials having cordate leaves and flowers a
rranged in conelike spikes; the dried flowers of this plant are used in b
rewing to add the characteristic bitter taste to beer
Synset('hop.n.03') an informal dance where popular music is played
Synset('hop.v.01') jump lightly
Synset('hop.v.02') move quickly from one place to another
Synset('hop.v.03') travel by means of an aircraft, bus, etc.
Synset('hop.v.04') traverse as if by a short airplane trip
Synset('hop.v.05') jump across
Synset('hop.v.06') make a jump forward or upward

Synsets for jump:
Synset('jump.n.01') a sudden and decisive increase
Synset('leap.n.02') an abrupt transition
Synset('jump.n.03') (film) an abrupt transition from one scene to another
Synset('startle.n.01') a sudden involuntary movement
Synset('jump.n.05') descent with a parachute
Synset('jump.n.06') the act of jumping; propelling yourself off the groun
d
Synset('jump.v.01') move forward by leaps and bounds
Synset('startle.v.02') move or jump suddenly, as if in surprise or alarm
Synset('jump.v.03') make a sudden physical attack on
Synset('jump.v.04') increase suddenly and significantly
Synset('leap_out.v.01') be highly noticeable
Synset('jump.v.06') enter eagerly into
Synset('rise.v.11') rise in rank or status
Synset('jump.v.08') jump down from an elevated point
Synset('derail.v.02') run off or leave the rails
Synset('chute.v.01') jump from an airplane and descend with a parachute
Synset('jump.v.11') cause to jump or leap
Synset('jumpstart.v.01') start (a car engine whose battery is dead) by co
nnecting it to another car's battery
Synset('jump.v.13') bypass
Synset('leap.v.02') pass abruptly from one state or topic to another
Synset('alternate.v.01') go back and forth; swing back and forth between
two states or conditions
Sentence used for Lesk: The rabbit hopped into the air.


Result of lesk: Synset('startle.n.01')

Definition: move or jump suddenly, as if in surprise or alarm
```

According to the Wu-Palmer algorithm the first verb definitions of hop and jump are quite similar, due to the show path distances between the two words. In addition, we can see that the synset with the most overlap between the sentence and the definitions of "Jump" is startle.n.01. This is a reasonable result, as a rabbit's hopping is usually sudden and without warning, and can be a result of being surprised.

SentiWordNet

SentiWordNet is a tool part of WordNet that is used to judge the "feeling" behind a word. This can be used to judge the emotion of a text, which may be useful in determining a persons reaction to what they are talking about or the current situation.

In [10]:

```
from nltk.corpus import sentiwordnet

# Find the synsets for a positive word
print(wordnet.synsets('Adore'))
print("\nDefinition:", wordnet.synset('adore.v.01').definition())

sentiAdore = sentiwordnet.senti_synset('adore.v.01')
print(sentiAdore)
print("Positive score = ", sentiAdore.pos_score())
print("Negative score = ", sentiAdore.neg_score())
print("Objective score = ", sentiAdore.obj_score())

print ("\nScore of each word in: Rabbits are impossibly adorable\n")
smallSentence = "Rabbits are impossibly adorable"
tokens = smallSentence.split()
for word in tokens:
    wordSent = list(sentiwordnet.senti_synsets(word))[0]
    print(wordSent)
    print("Positive score = ", wordSent.pos_score())
    print("Negative score = ", wordSent.neg_score())
    print("Objective score = ", wordSent.obj_score())
    print("\n")
```

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

Definition: love intensely
<adore.v.01: PosScore=0.5 NegScore=0.125>
Positive score =  0.5
Negative score =  0.125
Objective score =  0.375

Score of each word in: Rabbits are impossibly adorable

<rabbit.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0


<are.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0


<impossibly.r.01: PosScore=0.0 NegScore=0.125>
Positive score =  0.0
Negative score =  0.125
Objective score =  0.875


<adorable.s.01: PosScore=0.5 NegScore=0.0>
Positive score =  0.5
Negative score =  0.0
Objective score =  0.5
```

As we can see, the words that serve a more "neutral" and "practical" purpose have high objective scores but low positive and negative scores. In contrast, words like "impossibly" and "adorable" have lower objective scores and higher positive and negative scores. These scores are useful to have, as they can allow us to judge what the tone of a piece of text is.

Collocation

Collocations are when multiple words put together to give off a different (more often greater) meaning than they would when they are read individually. Examples include words like "Hard disk", "serving time", and "crystal clear".

Here are the collocations for text4 the Inaugural Address

In [11]:

```
from nltk.book import *

text4.collocations()
```

```
*** 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
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 of the collocation "Federal Government", we calculate this with the log of probability

In [12]:

```
import math
text = ' '.join(text4.tokens)
vocab = len(set(text4))
xy = text.count('Federal Government')/vocab
print("p(Federal Government) = ",xy )
x = text.count('Federal')/vocab
print("p(Federal) = ", x)
y = text.count('Government')/vocab
print('p(Government) = ', y)
pmi = math.log2(xy / (x * y))
print('PMI = ', pmi)
```

```
p(Federal Government) =  0.0031920199501246885
p(Federal) =  0.006483790523690773
p(Government) =  0.03371571072319202
PMI =  3.868067366919006
```

As we can see, the PMI value of "Federal Government" is roughly 3.87. This gives us a good indication that "Federal Government" is a collocation, and carries more mutual information.