Exploring WordNet

By: Aditi Chaudhari

WordNet is a huge database of English nouns, verbs, adjectives, and adverbs which have been grouped into synsets (cognitive synonyms) with each synset expressing a different meaning. These synsets are linked with semantic and lexical relations. Because of the way that WordNet is organized, it is a quite powerful tool for computational linguistics and NLP.

Setup

First, we need to import the wordnet package

In [42]:
```
import nltk
import ssl

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

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

```
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/aditichaudhari/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

The Noun Hierarchy in WordNet

Let's choose a noun and find all of its synsets

In [2]:
```
print(wn.synsets('woman'))
picked_synset_n = wn.synset('woman.n.01')

[Synset('woman.n.01'), Synset('woman.n.02'), Synset('charwoman.n.01'), Synset('womanhood.n.02')]
```

Now, let us take a look at the definition, an example of, and the lemmas of a the picked synset

In [3]:
```
print("definition of " + str(picked_synset_n) + ": " + str(picked_synset_n.definition()))
print("usage example of " + str(picked_synset_n) + ": " + str(picked_synset_n.examples()))
print("lemmas of " + str(picked_synset_n) + ": " + str(picked_synset_n.lemmas()))

definition of Synset('woman.n.01'): an adult female person (as opposed to a man)
usage example of Synset('woman.n.01'): ['the woman kept house while the man hunted']
lemmas of Synset('woman.n.01'): [Lemma('woman.n.01.woman'), Lemma('woman.n.01.adult_female')]
```

The synsets in WordNet are connected to other synsets in a hierarchical manner. Nouns in WordNet are defined in a hypernym-hyponym manner, where hypernyms are synsets that are higher in the hierarchy and hyponyms are synsets that are lower in the hierarchy. This segment of code traverses up the hierarchy until it reaches the very top: 'entity.n.01'. 'entity.n.01' is the top-most noun level.

In [4]:
```
hyp = picked_synset_n.hypernyms()[0]
top = wn.synset('entity.n.01')

while hyp:
    print(hyp)
    if hyp == top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]

Synset('adult.n.01')
Synset('person.n.01')
Synset('causal_agent.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Let's find the hypernyms, hyponyms, meronyms, holonyms, and antonyms of the selected synset.

In [5]:
```
print("\nhypernyms of " + str(picked_synset_n) + ": " + str(picked_synset_n.hypernyms()))
print("\nhyponyms of " + str(picked_synset_n) + ": " + str(picked_synset_n.hyponyms()))
print("\nmeronyms of " + str(picked_synset_n) + ": " + str(picked_synset_n.part_meronyms()))
print("\nholonyms of " + str(picked_synset_n) + ": " + str(picked_synset_n.part_holonyms()))
print("\nantonyms of " + str(picked_synset_n) + ": " + str(picked_synset_n.lemmas()[0].antonyms()))

hypernyms of Synset('woman.n.01'): [Synset('adult.n.01'), Synset('female.n.02')]

hyponyms of Synset('woman.n.01'): [Synset('amazon.n.01'), Synset('b-girl.n.01'), Synset('bachelor_girl.n.01'), Synset('baggage.n.02'), Synset('ball-buster.n.01'), Synset('black_woman.n.01'), Synset('bluestocking.n.01'), Synset('bridesmaid.n.01'), Synset('broad.n.01'), Synset('cat.n.03'), Synset('cinderella.n.01'), Synset('coquette.n.01'), Synset('dame.n.02'), Synset('debutante.n.01'), Synset('divorcee.n.01'), Synset('dominatrix.n.01'), Synset('donna.n.01'), Synset('enchantress.n.01'), Synset('ex-wife.n.01'), Synset('eyeful.n.01'), Synset('geisha.n.01'), Synset('girl.n.01'), Synset('girl.n.05'), Synset('girlfriend.n.01'), Synset('girlfriend.n.02'), Synset('gold_digger.n.02'), Synset('gravida.n.02'), Synset('heroine.n.02'), Synset('inamorata.n.01'), Synset('jezebel.n.02'), Synset('jilt.n.01'), Synset('lady.n.01'), Synset('maenad.n.01'), Synset('maenad.n.02'), Synset('matriarch.n.01'), Synset('matriarch.n.02'), Synset('matron.n.03'), Synset('mestiza.n.01'), Synset('mistress.n.01'), Synset('mother_figure.n.01'), Synset('nanny.n.01'), Synset('nullipara.n.01'), Synset('nymph.n.03'), Synset('nymphet.n.01'), Synset('old_woman.n.01'), Synset('prostitute.n.01'), Synset('shiksa.n.01'), Synset('smasher.n.02'), Synset('sylph.n.01'), Synset('unmarried_woman.n.01'), Synset('vestal.n.01'), Synset('wac.n.01'), Synset('wave.n.09'), Synset('white_woman.n.01'), Synset('widow.n.01'), Synset('wife.n.01'), Synset('wonder_woman.n.01'), Synset('yellow_woman.n.01')]

meronyms of Synset('woman.n.01'): [Synset('adult_female_body.n.01')]

holonyms of Synset('woman.n.01'): []

antonyms of Synset('woman.n.01'): [Lemma('man.n.01.man')]
```

The Verb Hierarchy in WordNet

Now, let's pick a verb and take a look at one of its synset's definition, usage example, and lemmas.

In [6]:
```
print(wn.synsets('love'))
picked_synset_v = wn.synset('love.v.03')

[Synset('love.n.01'), Synset('love.n.02'), Synset('beloved.n.01'), Synset('love.n.04'), Synset('love.n.05'), Synset('sexual_love.n.02'), Synset('love.v.01'), Synset('love.v.02'), Synset('love.v.03'), Synset('sleep_together.v.01')]
```

In [7]:
```
print("definition of " + str(picked_synset_v) + ": " + str(picked_synset_v.definition()))
print("usage example of " + str(picked_synset_v) + ": " + str(picked_synset_v.examples()))
print("lemmas of " + str(picked_synset_v) + ": " + str(picked_synset_v.lemmas()))

definition of Synset('love.v.03'): be enamored or in love with
usage example of Synset('love.v.03'): ['She loves her husband deeply']
lemmas of Synset('love.v.03'): [Lemma('love.v.03.love')]
```

Using the selected synset, we can traverse up the WordNet hierarchy as far as we can go. Like nouns, verbs can be organized in a hypernym-hyponym relationship, yet unlike nouns, verbs have no uniform top-level synset.

In [20]:
```
hyper = lambda s: s.hypernyms()
print(list(picked_synset_v.closure(hyper)))

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

Morphy

Using morphy, we can find different forms of the word.

In [15]:
```
print(wn.morphy('love', wn.VERB))

love
```

Similarity with the Wu-Palmer Similarity Metric and the Lesk Algorithm

Next, let us select two words and see how similar they are. Using the Wu-Palmer similarity metric between the nouns 'dog' and 'puppy', we can see that these two nouns are very similar. This makes sense because puppies are just juvenile dogs.

The lesk algorithm was also used in this segment of code to determine which meaning of 'puppy' was intended to be used in the sentence "I took my puppy to the dog park to play with the other dogs".

In [41]:
```
from nltk.wsd import lesk

dog = wn.synset('dog.n.01')
puppy = wn.synset('puppy.n.01')

print("The Wu-Palmer similarity metric between dog and puppy is: " + str(wn.wup_similarity(dog, puppy)))

sentence = ["I", "took", "my", "puppy", "to", "the", "dog", "park", "to", "play", "with", "the", "other", "dogs"]
print(str(lesk(sentence, 'puppy')) + " definition: " + lesk(sentence, 'puppy').definition())

The Wu-Palmer similarity metric between dog and puppy is: 0.8965517241379313
Synset('puppy.n.01') definition: a young dog
```

SentiWordNet

SentiWordNet is a resource for opinion mining. Every synset of WordNet is assigned to one of three sentiment categories: positive, negative, and objectivity. Using SentiWordNet, we can get a better understanding of the sentiment behind a word.

After playing with SentiWordNet, I am surprised that there is a slight positive score associated with the word 'hate'. SentiWordNet can be used to analyze the emotional meaning behind sentences and phrases, which has so many applications. For instance, sentiment analysis can be used for analyzing comments across social media to determine if you are building a good brand. It is important to keep in mind that sentiment analysis still has some limitations, though.

In [71]:
```
from nltk.corpus import sentiwordnet as swn

hate = list(swn.senti_synsets('hate'))
print(hate)

for item in hate:
    print(item)

sentence = ["I", "love", "nlp"]

print("\n")

for word in sentence:
    synsets = list(swn.senti_synsets(word))
    print("\n" + word)
    for item in synsets:
        print(item)

[SentiSynset('hate.n.01'), SentiSynset('hate.v.01')]
<hate.n.01: PosScore=0.125 NegScore=0.375>
<hate.v.01: PosScore=0.0 NegScore=0.75>



I
<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>
<one.s.01: PosScore=0.0 NegScore=0.25>

love
<love.n.01: PosScore=0.625 NegScore=0.0>
<love.n.02: PosScore=0.375 NegScore=0.0>
<beloved.n.01: PosScore=0.125 NegScore=0.0>
<love.n.04: PosScore=0.25 NegScore=0.0>
<love.n.05: PosScore=0.0 NegScore=0.0>
<sexual_love.n.02: PosScore=0.0 NegScore=0.0>
<love.v.01: PosScore=0.5 NegScore=0.0>
<love.v.02: PosScore=1.0 NegScore=0.0>
<love.v.03: PosScore=0.625 NegScore=0.0>
<sleep_together.v.01: PosScore=0.375 NegScore=0.125>

nlp
<natural_language_processing.n.01: PosScore=0.0 NegScore=0.0>
```

Collocations

Collocations are words that appear together who's whole meaning means more than each word individually. The collocation that I chose from text4 was 'United States', as 'United States' has a more significant meaning than 'United' or 'States' on their own. I calculated the mutual information, which outputted 4.82. Since the score is high, it is safe to assume it is likely to be a collocation.

In [72]:
```
from nltk.book import *
text4

*** 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
```

Out[72]:
```
<Text: Inaugural Address Corpus>
```

In [77]:
```
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
```

In [78]:
```
text = ' '.join(text4.tokens)
text[:50]
```

Out[78]:
```
'Fellow - Citizens of the Senate and of the House o'
```

In [83]:
```
import math

vocab = len(set(text4))
us = text.count('United States')/vocab
print("p(United States) =", us)
u = text.count('United')/vocab
print("p(United) =", u)
s = text.count('States')/vocab
print("p(States) =", s)
pmi = math.log2(us / (u * s))
print('pmi = ', pmi)
```

```
p(United States) = 0.015860349127182045
p(United) = 0.017057356608478
p(States) = 0.033017456359102244
pmi =  4.815657649020885
```

In [ ]: