What is WordNet?

WordNet is a lexical database of English that groups nouns, verbs, adjectives, and adverbs into sets of "cognitive synonyms (synsets). These synsets are interlinked by semantic and lexical relations, meaning that relations are built on more than meaning similarity. This structure makes WordNet especially useful for Natural Language Processing.

1. Let's import WordNet

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

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
```

2. Select a noun and output all synsets

```
synsets = wn.synsets('base')
synsets
```

```
[Synset('base.n.01'),
 Synset('foundation.n.03'),
 Synset('base.n.03'),
 Synset('base.n.04'),
 Synset('base.n.05'),
 Synset('floor.n.03'),
 Synset('basis.n.02'),
 Synset('base.n.08'),
 Synset('nucleotide.n.01'),
 Synset('base.n.10'),
 Synset('base.n.11'),
 Synset('basis.n.03'),
 Synset('base.n.13'),
 Synset('base.n.14'),
 Synset('al-qaeda.n.01'),
 Synset('root.n.03'),
 Synset('infrastructure.n.02'),
 Synset('base.n.18'),
 Synset('base.n.19'),
 Synset('base.n.20'),
 Synset('establish.v.08'),
 Synset('base.v.02'),
 Synset('free-base.v.01'),
 Synset('basal.s.02'),
 Synset('base.s.02'),
 Synset('base.s.03'),
 Synset('base.s.04'),
 Synset('base.s.05'),
 Synset('base.s.06'),
 Synset('base.s.07')]
```

3. Analyze a noun

```
base = wn.synset('base.n.01')
```

```
base.definition()
```

```
'installation from which a military force initiates operations'
```

```
base.examples()
```

```
['the attack wiped out our forward bases']
```

```
base.lemmas()
```

```
[Lemma('base.n.01.base'), Lemma('base.n.01.base_of_operations')]
```

```
# Traverse hierarchy
hypernym = base.hypernyms()[0]
top = wn.synset('entity.n.01')
while hypernym:
  print(hypernym)
  if hypernym == top:
    break
  if hypernym.hypernyms():
    hypernym = hypernym.hypernyms()[0]
```

```
Synset('military_installation.n.01')
Synset('facility.n.01')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Wordnet organizes nouns under a hierarchy that becomes more generalized as you traverse up, and specifics are left behind as you traverse up because they can only apply to that level. Important to note as well is the fact that nouns has a top-level synset, meaning that all nouns can be generalized to one noun.

"Base" was generalized under "military installation", and then that was generalized to "facility" - all the way up to "physical entity" and "entity" which encompass most and all nouns respectively.

4. Output relations

```
print("Hypernyms:")
base.hypernyms()
```

```
Hypernyms:
[Synset('military_installation.n.01')]
```

```
print("Hyponyms:")
base.hyponyms()
```

```
[Synset('air_base.n.01'),
 Synset('army_base.n.01'),
 Synset('firebase.n.01'),
 Synset('navy_base.n.01'),
 Synset('rocket_base.n.01')]
```

```
print("Meronyms:")
try:
  base.meronyms()
except:
  print([])
```

```
Meronyms:
[]
```

```
print("Holonyms:")
try:
  base.holonyms()
except:
  print([])
```

```
Holonyms:
[]
```

```
print("Antonyms:")
try:
  base.antonyms()
except:
  print([])
```

```
Antonyms:
[]
```

5. Select a verb and output all synsets

```
synsets = wn.synsets('run')
synsets
```

```
[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')]
```

6. Analyze a verb

```
run = wn.synset('run.v.01')
run
```

```
Synset('run.v.01')
```

```
run.definition()
```

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

```
run.examples()
```

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

```
run.lemmas()
```

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

```
# Traverse hierarchy
hypernym = run.hypernyms()[0]
top = wn.synset('travel.v.01')
while hypernym:
  print(hypernym)
  if hypernym == top:
      break
  if hypernym.hypernyms():
    hypernym = hypernym.hypernyms()[0]
```

```
Synset('travel_rapidly.v.01')
Synset('travel.v.01')
```

Wordnet also arranges verbs into hierarchies that generalize as you travel up. Verbs toward the bottom of hierarchies characterizes an event more specifically. Because of the broadness of verbs, verbs do not have a top-level synset.

This can be seen with run's hierarchy, which has a top-level of "travel" which does not describe many other actions.

7. Use morphy to find as many different forms of the word as you can

```
print(wn.morphy('run', wn.NOUN))
print(wn.morphy('run', wn.VERB))
print(wn.morphy('run', wn.ADJ))
```

```
run
run
None
```

8. Compare two words

```
synsets = wn.synsets('explode')
synsets
```

```
[Synset('explode.v.01'),
 Synset('explode.v.02'),
 Synset('explode.v.03'),
 Synset('explode.v.04'),
 Synset('explode.v.05'),
 Synset('explode.v.06'),
 Synset('explode.v.07'),
 Synset('explode.v.08'),
 Synset('detonate.v.02'),
 Synset('explode.v.10')]
```

```
explode = wn.synset('explode.v.05')
explode.definition()
```

```
'destroy by exploding'
```

```
# Lesk algorithm
from nltk.wsd import lesk
lesk(explode.definition().split(), 'destroy')
print(wn.synset('destroy.v.02').definition())
```

```
destroy completely; damage irreparably
```

```
synsets = wn.synsets('destroy')
synsets
```

```
[Synset('destroy.v.01'),
 Synset('destroy.v.02'),
 Synset('demolish.v.03'),
 Synset('destroy.v.04')]
```

```
# Get the synset for destroy that I think relates to explode the most
destroy = wn.synset('destroy.v.01')
destroy.definition()
```

```
'do away with, cause the destruction or undoing of'
```

```
# Wu-Palmer similarity metric for the first synset of destroy
wn.wup_similarity(explode, destroy)
```

```
0.3333333333333333
```

```
# Get the synset for destroy that the Lesk algorithm favors
destroy = wn.synset('destroy.v.02')
destroy.definition()
```

```
'destroy completely; damage irreparably'
```

```
# Wu-Palmer similarity metric for the second synset of destroy
wn.wup_similarity(explode, destroy)
```

```
0.4
```

Explode and destroy are loosely related. I think of destroying being a consequential action of exploding. The relation that I had imagined had a lower Wu-Palmer metric than the synset given by the Lesk algorithm. After, reevaluating, it makes sense when only considering defintions and the words contained in them why the other synset has a higher Wu-Palmer metric. The Wu-Palmer metric can only consider to a dictionary of glosses while I consider my experiences and relations built myself.

9. SentiWordNet

SentiWordNet is a resource that assigns sentiment scores to synsets. sentiment scores include positivity, negativity, and objectivity (neutral). SentiWordNet is useful for, as the name suggests, sentiment analysis. Some use cases include analyzing how a person is feeling in call menus and detecting emotional state for health.

```
# Setup
nltk.download('sentiwordnet')
from nltk.corpus import sentiwordnet as swn
```

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

```
# Select an emotionally charged word
# Find its senti-synsets and output polarity scores for each word

struggle = swn.senti_synset('contend.v.06')
print(struggle)
print("Positive score = ", struggle.pos_score())
print("Negative score = ", struggle.pos_score())
print("Objective score = ", struggle.obj_score())
```

```
<contend.v.06: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
```

```
# Make up a sentence
sen = "I think she doesn't want to be in a relationship because she is going to cheat on me again" # not related
# Output the polarity for each word in the sentence
for word in sen.split():
  synsets = list(swn.senti_synsets(word))
  if synsets:
    synset = synsets[0]
    print("Word:", word, "\nPos: ", synset.pos_score(), " - Neg: ", synset.neg_score(), " - Obj: ", synset.obj_score(), '\n')
# Write a couple of sentences about your observations of the scores and the utility of knowing these scores in an NLP applications
```

```
Word: I
Pos:  0.0  - Neg:  0.0  - Obj:  1.0
```

```
Word: think
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: want
Pos:  0.0  - Neg:  0.25  - Obj:  0.75

Word: be
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: in
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: a
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: relationship
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: is
Pos:  0.25  - Neg:  0.125  - Obj:  0.625

Word: going
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: cheat
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: on
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: me
Pos:  0.0  - Neg:  0.0  - Obj:  1.0

Word: again
Pos:  0.0  - Neg:  0.0  - Obj:  1.0
```

The sentiment score is hard to find, given the in-class demonstration and my experiences with this assignment, it seems that there are a lot of words that don't carry objective sentiment. I assume that this means that SentiWordNet doesn't know what to make of these words. It is also evident that many words are objectively ranked because context within the sentence and relative to other words is not considered.

The drawbacks of this are not necessarily bad. This type of analysis allows for "objective" sentiment analysis that is unbothered by potentially skewing characteristics.

Sentiment scores in NLP are extremely useful as they can connect the flat and lifeless definitions that are available through WordNet with the more human aspects of language, such as emotion and colloquialisms. Greater insight for computers in how humans approach langugage allows NLP to be more accurate.

10. Collocation

Collocations are comibinations of words that take different meaning than they would individually, and there is not a word that has the same meaning. Common collocations include

- Catch a cold
- Good enough
- Take a look
- Dead ahead

```
# Collocations for the inagural corpus
from nltk.book import text4
text4.collocation_list()
```

```
[('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')]
```

```
# Calculate mutual information for a collocation
import math
l = len(text4.tokens)
x = text4.count('Almighty')
y = text4.count('God')

numerator = min(x,y) / l
denominator = (x/l) * (y/l)

numerator
```

```
math.log( numerator / denominator , 2)
```

```
10.427822450300944
```

Because the PMI is over 0, Almighty God is a collocation. This makes some sense when considering the statistic. "Almighty" and "God" rarely appear without each other in normal text, since their meanings are highly related. However, they are a collocation because their meaning together changes the way God is perceived. God as a word does not imply almighty.