1 - WordNet

WordNet is a database of words (which includes nouns, adjectives, and adverbs) as well as their defintions, synonyms, antonyms, definition, examples, lemmas, and others. The purpose of this database is to group words using some sort of hierarchy system.

```
# 2
import nltk
nltk.download('omw-1.4')
nltk.download('wordnet')
from nltk.corpus import wordnet as wn
wn.synsets('arm')
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package wordnet to /root/nltk_data...
[Synset('arm.n.01'),
 Synset('arm.n.02'),
 Synset('weapon.n.01'),
 Synset('arm.n.04'),
 Synset('branch.n.01'),
 Synset('sleeve.n.01'),
 Synset('arm.v.01'),
 Synset('arm.v.02')]
```

```
# 3
print(wn.synset('sleeve.n.01').definition())
print(wn.synset('sleeve.n.01').examples())
print(wn.synset('sleeve.n.01').lemmas())

# traverse up
sleeve = wn.synset('sleeve.n.01')
hyper = lambda s: s.hypernyms()
list(sleeve.closure(hyper))
```

```
the part of a garment that is attached at the armhole and that provides a cloth c
[]
[Lemma('sleeve.n.01.sleeve'), Lemma('sleeve.n.01.arm')]
[Synset('cloth_covering.n.01'),
 Synset('covering.n.02'),
 Synset('artifact.n.01'),
 Synset('whole.n.02'),
 Synset('object.n.01'),
 Synset('physical_entity.n.01'),
 Synset('entity.n.01')]
```

One observation that I have made that it seems like the hierarchy of most words starts from the word 'entity'. As we move up the hierarhcy, the words become less general.

```
# 4
print(wn.synset('sleeve.n.01').hypernyms())
print(wn.synset('sleeve.n.01').hyponyms())
print(wn.synset('sleeve.n.01').part_meronyms())
print(wn.synset('sleeve.n.01').part_holonyms())

sleeve = wn.synsets('sleeve', pos=wn.ADJ)
print(sleeve) # no antonym exists for the word "sleeve"
```

```
[Synset('cloth_covering.n.01')]
[Synset('dolman_sleeve.n.01'), Synset('long_sleeve.n.01'), Synset('raglan_sleeve
[Synset('cuff.n.01'), Synset('elbow.n.04'), Synset('wristband.n.01')]
[Synset('garment.n.01')]
[]
```

```
# 5
wn.synsets('run')
```

```
[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
print(wn.synset('move.v.13').definition())
print(wn.synset('move.v.13').examples())
print(wn.synset('move.v.13').lemmas())

# traverse up
move = wn.synset('move.v.13')
hyper = lambda s: s.hypernyms()
list(move.closure(hyper))
```

```
progress by being changed
['The speech has to go through several more drafts', 'run through your presentat
[Lemma('move.v.13.move'), Lemma('move.v.13.go'), Lemma('move.v.13.run')]
[Synset('change.v.02')]
```

It seems that it's a little bit more specific compared to the way nouns are organized. There's no a common word at the beginning of the hierarchy.

```
# 7
wn.morphy('move', wn.VERB)
```

```
    'move'

# 8
move = wn.synset('move.v.13')
run = wn.synset('run.v.32')
print(wn.path_similarity(move, run))

    0.16666666666666666
```

I chose the words "run" and "move". Even though those words are very similar to me, the similarity is only .166 which is much lower than I expected.

9 - SentiWordNet

SentiWordNet is a tool that was built on WordNet that assigns 3 scores (positivity, negativity, and objectivity) for each sysnset. All 3 scores always add up to 1. This tool is used for finding out wheather the sentence is mostly positive, negative, or neutral.

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

synsets = wn.synsets('love')
for s in synsets:
  breakdown = swn.senti_synset(s.name())
  print(breakdown)
  print("Positive score = ", breakdown.pos_score())
  print("Negative score = ", breakdown.neg_score())
  print("Objective score = ", breakdown.obj_score())

print('\n\n')
sentence = 'I love playing ping pong!!!'
neg = 0
pos = 0
tokens = sentence.split()
for token in tokens:
  syn_list = list(swn.senti_synsets(token))
  if syn_list:
        syn = syn_list[0]
        neg += syn.neg_score()
        pos += syn.pos_score()
        print(token, '\tneg:', syn.neg_score(), 'pos', syn.pos_score())
print('The total score of the sentece:')
print('neg\tpos')
print(neg, '\t', pos)
```

```
<love.n.01: PosScore=0.625 NegScore=0.0>
Positive score =  0.625
Negative score =  0.0
Objective score =  0.375
<love.n.02: PosScore=0.375 NegScore=0.0>
Positive score =  0.375
Negative score =  0.0
Objective score =  0.625
<beloved.n.01: PosScore=0.125 NegScore=0.0>
Positive score =  0.125
Negative score =  0.0
Objective score =  0.875
<love.n.04: PosScore=0.25 NegScore=0.0>
Positive score =  0.25
Negative score =  0.0
Objective score =  0.75
<love.n.05: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
<sexual_love.n.02: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
<love.v.01: PosScore=0.5 NegScore=0.0>
Positive score =  0.5
Negative score =  0.0
Objective score =  0.5
<love.v.02: PosScore=1.0 NegScore=0.0>
Positive score =  1.0
Negative score =  0.0
Objective score =  0.0
<love.v.03: PosScore=0.625 NegScore=0.0>
Positive score =  0.625
Negative score =  0.0
Objective score =  0.375
<sleep_together.v.01: PosScore=0.375 NegScore=0.125>
Positive score =  0.375
Negative score =  0.125
Objective score =  0.5
```

```
I       neg: 0.0 pos 0.0
love    neg: 0.0 pos 0.625
playing         neg: 0.0 pos 0.0
ping    neg: 0.0 pos 0.0
The total score of the sentece:
neg     pos
0.0      0.625
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
```

SentiWordNet Observations:

I would say the scores for the most part are quite accurate. Alothough I do not agree with all of them. For example, the word "beloved" received .875 for objectivity and .125 for positivity. I expected that the positivity score would be a lot higher. I also expected a higher positivity score for my sentense because I used a quite positive word "love" as well as added 3 exclamation points for more emphasis. The sentence got .625 for positivity, however, I expected that it would be around .9. Perhaps there's some bias from my end as I believe that the word 'love' is the most positive word ever. But I would imagine some people would disagree with me, so perhaps the program does a better job of objectively calculating how something is positive, negative, or netural.

SentiWordNet could be used in many circumstances. For example, it could be used to find out wheather a review left on a website is positive or negative. Another example example is finding out wheather there's bias in a news media or a magazine by calculating how much of their vocabulary is objective

Double-click (or enter) to edit

10 - Collocations

Collocatinos is when there's a greater probability of one word appearing after another word. For example, if you see the word "thank", there's a high probability that next word is 'you'. When there's a collocation, 2 words combined have more meaning compared to when they are separate.

```
nltk.download('book')
from nltk.book import *
import math
text4.collocations()
```

```
[nltk_data] Downloading collection 'book'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
```

```
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package ieer to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ieer.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package nps_chat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/nps_chat.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package ppattach to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/ppattach.zip.
[nltk_data]    | Downloading package reuters to /root/nltk_data...
[nltk_data]    | Downloading package senseval to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/senseval.zip.
[nltk_data]    | Downloading package state_union to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/state_union.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package swadesh to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/swadesh.zip.
[nltk_data]    | Downloading package timit to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/timit.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package toolbox to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/toolbox.zip.
[nltk_data]    | Downloading package udhr to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr.zip.
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/udhr2.zip.
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/unicode_samples.zip.
[nltk_data]    | Downloading package webtext to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/webtext.zip.
```

```
text = ' '.join(text4.tokens)
vocab = len(set(text4))
fy = text.count('four years') / vocab
print('p(four years)', fy)
f = text.count('four') / vocab
print('p(four)', f)
y = text.count('years') / vocab
print('p(years)', y)
pmi = math.log2(fy / (f * y))
print('pmi: ', pmi)
```

```
p(four years) 0.0024937655860349127
p(four) 0.0035910224438902745
```

```
p(years) 0.014264339152119701
pmi:  5.605374467783668
```

Commentary on the results of the mutual information formula

Since the pmi is positive, it means that it's likely a collocation. Thus, the words "four" and "years" often times do go together according to their frequences found in text4. It aslo means that the words "four" and "years" generally do give more meaning compared to when they are separate.