WordNet

WordNet is a hierarchal organization of words that helps us get a listing of relations to other words. In essence it acts almost like a large dictionary and thesaurus where it can help us with certain semantic relations and organizations of words. It can be used to find many things such as synsents, hypernyms, antonyms, hierarchy, root words, and many other semantical and lexical relations that lie within words. It is a really powerful tool and can be used greatly in NLP projects to further understand certain words and contextual clues.

```
from nltk.corpus import wordnet as wn
```

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

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

```
wn.synsets('machine')
```

```
[Synset('machine.n.01'),
 Synset('machine.n.02'),
 Synset('machine.n.03'),
 Synset('machine.n.04'),
 Synset('machine.n.05'),
 Synset('car.n.01'),
 Synset('machine.v.01'),
 Synset('machine.v.02')]
```

```
wn.synset('machine.n.02').definition()
```

```
'an efficient person'
```

```
wn.synset('machine.n.02').examples()
```

```
['the boxer was a magnificent fighting machine']
```

```
wn.synset('machine.n.02').lemmas()
```

```
[Lemma('machine.n.02.machine')]
```

```
machine = wn.synset('machine.n.02')
hyper = lambda s: s.hypernyms()
list(machine.closure(hyper))
```

```
/usr/local/lib/python3.8/dist-packages/nltk/corpus/reader/wordnet.py:599: UserWarning: Discarded redundant search for Synset
  for synset in acyclic_breadth_first(self, rel, depth):
[Synset('person.n.01'),
 Synset('causal_agent.n.01'),
 Synset('organism.n.01'),
 Synset('physical_entity.n.01'),
 Synset('living_thing.n.01'),
 Synset('entity.n.01'),
 Synset('whole.n.02'),
 Synset('object.n.01')]
```

Hierarchy of Nouns

When looking at the hierarchy of nouns from the word machine, we see that the final synset on which it ends is the word object. When looking at the definition of the word noun we can see that in essence it boils down to a person place or object so it makes sense that the final synset would end on object as that as a far encompassing word that can take on a hold of many things. The most interesting thing though to make note of though is that nouns have a very large hierarchial tree as each specific object can be classified into broader and broader terms until you finally land on object, so if a different word than machine were to be used the first few iterations would be different. However, towards the end the last few iterations would be very similar to many other words as entity, whole, and object can be said about many things.

```
wn.synset('machine.n.02').hyponyms()
```

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

```
wn.synset('machine.n.02').hypernyms()
```

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

```
machine.lemmas()[0].antonyms()
```

```
[]
```

```
wn.synset('machine.n.02').part_meronyms()
```

```
[]
```

```
wn.synset('machine.n.02').part_holonyms()
```

```
[]
```

Using A Verb With Synset

```
wn.synsets('throw')
```

```
[Synset('throw.n.01'),
 Synset('throw.n.02'),
 Synset('throw.n.03'),
 Synset('throw.n.04'),
 Synset('throw.n.05'),
 Synset('throw.v.01'),
 Synset('throw.v.02'),
 Synset('shed.v.01'),
 Synset('throw.v.04'),
 Synset('give.v.07'),
 Synset('throw.v.06'),
 Synset('project.v.10'),
 Synset('throw.v.08'),
 Synset('bewilder.v.02'),
 Synset('hurl.v.03'),
 Synset('hold.v.03'),
 Synset('throw.v.12'),
 Synset('throw.v.13'),
 Synset('throw.v.14'),
 Synset('confuse.v.02')]
```

```
wn.synset('throw.v.01').definition()
```

```
'propel through the air'
```

```
wn.synset('throw.v.01').examples()
```

```
['throw a frisbee']
```

```
wn.synset('throw.v.01').lemmas()
```

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

```
throw = wn.synset('throw.v.01')
hyper = lambda s: s.hypernyms()
list(throw.closure(hyper))
```

```
[Synset('propel.v.01'), Synset('move.v.02')]
```

Hierarchy of Verbs

When looking at the hierarchy of verbs it is extremely different from that of nouns. For starters it seems that for verbs the hierarchy is much smaller than that of nouns and it makes sense that this is the case. For verbs it is not very common for their to be a large over arching word that can encompass many verbs in the way that for nouns there are words that have an overarching reach on many words. It does seem though that for verbs they do belong to a body of movement it seems in the aspect that throw belongs to move as would a word like run belong to move however it just reaches that word in a much shorter distance.

```
print(wn.morphy('throw'))
print(wn.morphy('threw'))
print(wn.morphy('throwing'))
print(wn.morphy('throws'))
```

```
throw
throw
throw
throw
```

Checking Similarity of Two Words

```
wn.synsets('drum')
```

```
[Synset('drum.n.01'),
 Synset('drum.n.02'),
 Synset('barrel.n.03'),
 Synset('drum.n.04'),
 Synset('brake_drum.n.01'),
 Synset('drum.n.06'),
 Synset('drum.v.01'),
 Synset('drum.v.02'),
 Synset('cram.v.03')]
```

```
wn.synsets('sing')
```

```
[Synset('sing.v.01'),
 Synset('sing.v.02'),
 Synset('sing.v.03'),
 Synset('whistle.v.05'),
 Synset('spill_the_beans.v.01')]
```

```
drum = wn.synset('drum.n.01')
sing = wn.synset('sing.v.01')
wn.wup_similarity(drum, sing)
```

```
0.1111111111111111
```

```
from nltk.wsd import lesk
```

```
for ss in wn.synsets('drum'):
    print(ss, ss.definition())
```

```
Synset('drum.n.01') a musical percussion instrument; usually consists of a hollow cylinder with a membrane stretched across
Synset('drum.n.02') the sound of a drum
Synset('barrel.n.03') a bulging cylindrical shape; hollow with flat ends
Synset('drum.n.04') a cylindrical metal container used for shipping or storage of liquids
Synset('brake_drum.n.01') a hollow cast-iron cylinder attached to the wheel that forms part of the brakes
Synset('drum.n.06') small to medium-sized bottom-dwelling food and game fishes of shallow coastal and fresh waters that make
Synset('drum.v.01') make a rhythmic sound
Synset('drum.v.02') play a percussion instrument
Synset('cram.v.03') study intensively, as before an exam
```

```
sentence = ['The', 'way', 'he','was','able','to', 'drum','the', 'instrument' ,'was','interesting']
print(lesk(sentence, 'drum'))
```

```
Synset('drum.n.02')
```

```
for ss in wn.synsets('sing'):
    print(ss, ss.definition())
```

```
Synset('sing.v.01') deliver by singing
Synset('sing.v.02') produce tones with the voice
Synset('sing.v.03') to make melodious sounds
Synset('whistle.v.05') make a whining, ringing, or whistling sound
Synset('spill_the_beans.v.01') divulge confidential information or secrets
```

```
sentence2 = ['the', 'music', 'teacher', 'made', 'the', 'child', 'sing']
print(lesk(sentence, 'sing'))
```

```
Synset('sing.v.03')
```

Lesk and Wu Palmer Similarity

When using the Wu Palmer similarity, the main thing that was noticeable is that the Wu Palmer looks at the similarity between the two chosen words and sees where their ancestors depth is to see how much of a similarity there is. With the Lesk algorithm, it mainly looks at the overlap and context of words to see what use of the specific word matches best with the intended use of the word. When using the Wu Palmer to compare the two words drum and sing, the interesting thing was that their similarity score was only 0.111. I was assuming that since they both come from a musical origin, their score would be higher, but in reality the score was much lower. This may be due to the specific use of the word in the sense that drum may be used as a noun and sing may be used as a verb which may cause some sense of difference. When running the Lesk algorithm on the word sing it was able to do a fairly great job in discerning which version of sing was the correct one utilizing the context. What was interesting though was when I removed the word music from the sentence the algorithm was still able to discern the correct use still meaning that some different uses of context and rules were used. When running the lesk algorithm on drum however, it gave back the use of drum as a noun rather than the intended meaning of it as a verb. This may be due to the context of instrument being around it casuing it to use the noun version. When instrument was removed though, it changed to a different noun that was even further off what it should have been.

SentiWordNet

SentiWordNet is a tool in nltk that allows for opinion mining and getting emotions and feelings in the form of a score to allow for the ability to discern if a statement or word may be in a positive or negative connotation. The tool returns a score of 0 to 1.0 in both the negative and positive category. If a score has a more positive tone then the positive score will be higher than the negative and vice versa. There are many uses cases for sentiment analysis and SentiWordNet such as using it to detect cyberbullying online with comments and tweets, the prediction of how people feel about stocks, how people feel about certain movies, and many other uses.

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

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

```
senti_list = list(swn.senti_synsets('love'))
for item in senti_list:
    print(item)
```

```
<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>
```

```
sent = 'I love that person with all my heart'
neg = 0
pos = 0
tokens = sent.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('neg score for',token , ":",  syn.neg_score())
        print('pos score for',token,":",  syn.pos_score())
print("added up score")
print("neg\tpos counts")
print(neg, '\t', pos)
```

```
neg score for I : 0.0
pos score for I : 0.0
neg score for love : 0.0
pos score for love : 0.625
neg score for person : 0.0
pos score for person : 0.0
neg score for all : 0.0
pos score for all : 0.0
```

```
neg score for heart : 0.125
pos score for heart : 0.0
added up score
neg     pos counts
0.125    0.625
```

SentiWordNet Scores and Output

When looking at the scores for the sentisynset of love, we can see that a majority of the scores are positive with love.v.03 being the highest positive score as that is the one with the definition of being enamored with. The score that was interesting was the synset for sleeping together which came back with a positive score of 0.375 and a negative score of 0.125. The negative score could be due to the aspect of the connotation of sleeping together in the public eye is still viewed as a taboo, so this could explain as to why there is a negative score associated with it based off of previous training data the SentiWordNet may have received. When looking at the SentiWordNet score for the use of a sentence we see that it is indeed a positive sentence, but the negative score for the token heart returns back negative. This could be due to the context surrounding word, or heart may have already have a negative score on its own being. This was an interesting output when taking into the account how emotionally charged the sentence was in the positive aspect.

Collocations

A collocation is when two words combine together to have a specific meaning that when substituted with other words would not make sense. For example the saying 'sweet ride' means that someones car is nice and new. The word sweet however, can not be replaced by something like sugary because the phrase would not mean the same. Collocations are formed when a specific word appears the chance that the next word appeears is very high as well.

```
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
from nltk.book import *
nltk.download('stopwords')
```

```
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Package treebank is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
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
```

```
text = ' '.join(text4.tokens)
import math
vocab = len(set(text6))
hg = text.count('Federal Government')/vocab
print("p(Federal Government) = ",hg )
h = text.count('Federal')/vocab
print("p(Federal) = ", h)
g = text.count('Government')/vocab
print('p(Government) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
p(Federal Government) =  0.014773776546629732
p(Federal) =  0.030009233610341645
```

```
p(Government) =  0.15604801477377656
pmi =  1.6575702782976882
```

Mutual Information

After calculating mutual information we see that the collocation Federal Government gets the score of 1.65 meaning that it has a fairly high chance of being a collocation. The higher the score of the pmi, the more likely a phrase is to be a collocation. My interpretation of mutual information is that a specific value is dependant on the value before it given pieces of data. For example, we see that Federal Government got a pmi score of 1.65 meaning that if the word Federal shows up, then there is a fairly good chance the next word to show up would be Government because of the fact that they are a collocation. If the score was lower, this would mean that there are more chances that other words can show up after Federal or before Government.