Jesse Truong JTT190006 9/22/2022 WordNets

WordNet is a database for the English Lanaguage tailor NLP. Wordnet its based on synset which are oranized in a hierachichal relations. WordNet connects words into relations like hyponyms, synonyms, and meronyms. This also helps the find words that mean the same thing and that are in the hierachial.

```
import nltk
nltk.download('omw-1.4')
nltk.download('wordnet')
nltk.download('sentiwordnet')
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('stopwords')
from nltk.corpus import wordnet as wn
from nltk.book import *
```

```
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
[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]   Unzipping corpora/stopwords.zip.
```

```
noun = wn.synsets('house')
print(noun)
```

✓ 0s completed at 3:42 AM

```
[Synset('house.n.01'), Synset('firm.n.01'), Synset('house.n.03'), Synset('house.n.04'
```

```
print("\nDefination: "+wn.synset('house.n.01').definition())
print("\nExamples: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').examples()]
print("\nLemmas: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').lemmas()]
curr = wn.synset('house.n.01')
top = wn.synset('entity.n.01')
print("\nHierarchy")
while curr:
  print(curr)
  if curr.hypernyms():
    curr = curr.hypernyms()[0]
  if curr == top:
    break
```

```
Defination: a dwelling that serves as living quarters for one or more families

Examples:
he has a house on Cape Cod
she felt she had to get out of the house

Lemmas:
Lemma('house.n.01.house')

Hierarchy
Synset('house.n.01')
Synset('building.n.01')
Synset('structure.n.01')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
```

WordNet Noun hierarchy organization is based of generality, the higher the word is on the hierarchy, the more general it is, and visa versa for the floor. The most general Noun is always entity.n01. For the noun, the hierarchys goes from house, the most specific noun at the bottom, going to more general noun as we travel up the hierachy, next building, structure, etc, all the way up to the top of the hierachry, entity.

```
print("\nHypernyms: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').hypernyms()]
print("\nHyponyms: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').hyponyms()]
```

```
print("\nMeronyms: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').part_meronyms() ]
print("\nHolonyms: ")
[print(i, end= "\n") for i in wn.synset('house.n.01').part_holonyms() ]
print("\nAntonyms: ")
[print(i, end= "\n") for i in wn.synsets('house.n.01', pos=wn.ADJ)]
```

```
Hypernyms:
Synset('building.n.01')
Synset('dwelling.n.01')

Hyponyms:
Synset('beach_house.n.01')
Synset('boarding_house.n.01')
Synset('bungalow.n.01')
Synset('cabin.n.02')
Synset('chalet.n.01')
Synset('chapterhouse.n.02')
Synset('country_house.n.01')
Synset('detached_house.n.01')
Synset('dollhouse.n.01')
Synset('duplex_house.n.01')
Synset('farmhouse.n.01')
Synset('gatehouse.n.01')
Synset('guesthouse.n.01')
Synset('hacienda.n.02')
Synset('lodge.n.04')
Synset('lodging_house.n.01')
Synset('maisonette.n.02')
Synset('mansion.n.02')
Synset('ranch_house.n.01')
Synset('residence.n.02')
Synset('row_house.n.01')
Synset('safe_house.n.01')
Synset('saltbox.n.01')
Synset('sod_house.n.01')
Synset('solar_house.n.01')
Synset('tract_house.n.01')
Synset('villa.n.02')

Meronyms:
Synset('library.n.01')
Synset('loft.n.02')
Synset('porch.n.01')
Synset('study.n.05')

Holonyms:

Antonyms:
[]
```

```
verb = wn.synsets('running')
print(verb)
```

```
[Synset('run.n.05'), Synset('run.n.07'), Synset('running.n.03'), Synset('running.n.04
```

```
print("\nDefination: "+wn.synset('scat.v.01').definition())
print("\nExamples: ")
[print(i, end= "\n") for i in wn.synset('scat.v.01').examples()]
print("\nLemmas: ")
[print(i, end= "\n") for i in wn.synset('scat.v.01').lemmas()]
curr = wn.synset('scat.v.01')
print("\nHierarchy")
hyper = lambda s: s.hypernyms()
list(curr.closure(hyper))
```

WordNet Verbs hierarchy organization is based of generality, the higher the word is on the hierarchy, the more general it is, and visa versa for the floor. Unlike the Noun, verbs have a different general at the hierarchy verbs instead like noun where every noun connects back to entity. So for the verb Scat, the hierachy goes to scat, then to travel, as there is no more general verb than travel.

```
wn.morphy('scat.v.01', wn.ADV)
wn.morphy('scat.v.01', wn.ADJ)
wn.morphy('scat.v.01', wn.NOUN)
wn.morphy('scat.v.01')
```

```
curr = wn.synset('slaughter.v.01')
hyper = lambda s: s.hypernyms()
print(list(curr.closure(hyper)))
kill = wn.synset('kill.v.01')
slaughter = wn.synset('slaughter.v.01')
print(wn.path_similarity(kill, slaughter))

print("Wu Similarity: "+str(wn.wup_similarity(kill, slaughter)))
```

```
from nltk.wsd import lesk
for ss in wn.synsets('club'):
    print(ss, ss.definition())
club = wn.synsets("club")
print(club)
sent = ['I', 'hit', 'a','ball','in','one','with','my','golf','club','.']
print(lesk(sent, 'club'))
```

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

```
0.5
Wu Similarity: 0.6666666666666666
Synset('baseball_club.n.01') a team of professional baseball players who play and tra
Synset('club.n.02') a formal association of people with similar interests
Synset('club.n.03') stout stick that is larger at one end
Synset('clubhouse.n.01') a building that is occupied by a social club
Synset('golf_club.n.02') golf equipment used by a golfer to hit a golf ball
Synset('club.n.06') a playing card in the minor suit that has one or more black trefo
Synset('cabaret.n.01') a spot that is open late at night and that provides entertainm
Synset('club.v.01') unite with a common purpose
Synset('club.v.02') gather and spend time together
Synset('club.v.03') strike with a club or a bludgeon
Synset('club.v.04') gather into a club-like mass
[Synset('baseball_club.n.01'), Synset('club.n.02'), Synset('club.n.03'), Synset('club
Synset('golf_club.n.02')
```

Similarity is based on the if a word is in a common anecestor, in this case, slaughter is based on the anecestor kill. The lesk algorthim is based on overlapping words in the defination of the words. In the term of our case, golf is overlapping withe the defination as well as ball.

SentiWordNet is based on the opinion mining concept. Sentances can be based on 3 sentiment scores: positivity, negativity, and objectivity. WordNet are score based on these 3 scores.

```
from nltk.corpus import sentiwordnet as swn

breakdown = swn.senti_synset('stupid.n.01')
print(breakdown)
print("Positive = ", breakdown.pos_score())
print("Negative = ", breakdown.neg_score())
print("Objective = ", breakdown.obj_score())

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

sent = ['Great', 'job', 'today','!','Super','happy','with','the','work','you','have','done
for token in sent:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        print(str(token) + "\tPositive = ", str(syn.pos_score())+ "\t\tNegative = ", str(sy
```

```
<stupid.n.01: PosScore=0.0 NegScore=0.125>
Positive =  0.0
Negative =  0.125
Objective =  0.875
<stupid.n.01: PosScore=0.0 NegScore=0.125>
```

```
<stupid.a.01: PosScore=0.0 NegScore=0.75>
<dazed.s.01: PosScore=0.0 NegScore=0.125>
<unintelligent.a.01: PosScore=0.0 NegScore=0.375>
Great   Positive =  0.0         Negative =  0.0
job     Positive =  0.0         Negative =  0.0
today   Positive =  0.125               Negative =  0.0
Super   Positive =  0.0         Negative =  0.0
happy   Positive =  0.875               Negative =  0.0
work    Positive =  0.0         Negative =  0.0
have    Positive =  0.0         Negative =  0.0
done    Positive =  0.0         Negative =  0.0
```

For the senti-synsets words, it seems its more of synonym of the word, which makes sense. And have a some whats similar score. In the sent, suprisingly great doesn't have any positive score as well as super. But words like today have a positive score, which is surpisingly. But in total the sentance has a positive score of 1 and a negative score of 0.

Collocations was when words combine to make a sum of their meanings. For example close, means 'a short distance away ' and together, 'with or in proximity to another person' so close together would be combine to make a great meaning.

```
print(text4.collocations())
text = ' '.join(text4.tokens)

import math
vocab = len(set(text6))
hg = text.count('fellow citizens')/vocab
print("p(fellow citizens) = ",hg )
h = text.count('fellow')/vocab
print("p(fellow) = ", h)
g = text.count('citizens')/vocab
print('p(citizens) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
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
None
p(fellow citizens) =  0.02816251154201293
p(fellow) =  0.06325023084025855
p(citizens) =  0.12465373961218837
pmi =  1.8367071851598558
```

We can see the pmi for fellow citizen 1.8367 which is good as a PMI that is positive which

We can see the pmi for fellow citizen 1.8367, which is good as a PMI that is positive, which means its likely to be a collocation. Taking the p of fellow citizen in the text, and taking the indival prop of fellow and citzen would take the estimane the amount of times the words was used indiviaully and when it was used together. Giving the propaiblity of the collocation and its indivual.