wordNet

February 25, 2023

1 WordNet

1.0.1 Summary

WordNet is something akin to a dictionary. It is a database that houses nouns, adjectives, verbs and adverbs that all come with something called 'glosses'. These are short definitions of the word. Additionally, WordNet also provides synsets, a set of synonyms for a certain word. WordNet has it's own hierarchical design due to it's beginnings as a way to support theroies over human semantic memory.

[ ]:
```
#import wordnet and select a noun. Output all sysnsets
from nltk.corpus import wordnet as wn
wn.synsets('house')
```

[ ]:
```
[Synset('house.n.01'),
 Synset('firm.n.01'),
 Synset('house.n.03'),
 Synset('house.n.04'),
 Synset('house.n.05'),
 Synset('house.n.06'),
 Synset('house.n.07'),
 Synset('sign_of_the_zodiac.n.01'),
 Synset('house.n.09'),
 Synset('family.n.01'),
 Synset('theater.n.01'),
 Synset('house.n.12'),
 Synset('house.v.01'),
 Synset('house.v.02')]
```

[ ]:
```
#select a synset and go up the hierarchy
noun = wn.synset('house.n.09')
print(noun)
print(noun.definition(),
print(noun.lemmas()),
print(noun.examples()))
print("\n")

print("Hypernyms of House (casino): ", )
```

```
hypernyms = noun.hypernyms()
while hypernyms:
    print(hypernyms[0].name())
    hypernyms = hypernyms[0].hypernyms()

print("Hyponyms of House (casino): ", )
hyponyms = noun.hyponyms()
for hyponym in hyponyms:
    print(hyponym.name())

print("Meronyms of House (casino): ", )
meronyms = noun.part_meronyms()
for meronym in meronyms:
    print(meronym.name())

print("Holonyms of House (casino): ", )
holonyms = noun.part_holonyms()
for holonym in holonyms:
    print(holonym.name())

print("Antonyms of House (casino): ", )
#lemmatize
lemmas = noun.lemmas()
for lemma in lemmas:
    antonyms = lemma.antonyms()
    if antonyms:
        for antonym in antonyms:
            print(antonym.name())



wn.synsets('good')
```

```
Synset('house.n.09')
[Lemma('house.n.09.house')]
['the house gets a percentage of every bet']
the management of a gambling house or casino None None


Hypernyms of House (casino):
management.n.02
administration.n.02
body.n.02
social_group.n.01
group.n.01
abstraction.n.06
entity.n.01
Hyponyms of House (casino):
```

```
Meronyms of House (casino):
Holonyms of House (casino):
Antonyms of House (casino):
```

[ ]:
```
[Synset('good.n.01'),
 Synset('good.n.02'),
 Synset('good.n.03'),
 Synset('commodity.n.01'),
 Synset('good.a.01'),
 Synset('full.s.06'),
 Synset('good.a.03'),
 Synset('estimable.s.02'),
 Synset('beneficial.s.01'),
 Synset('good.s.06'),
 Synset('good.s.07'),
 Synset('adept.s.01'),
 Synset('good.s.09'),
 Synset('dear.s.02'),
 Synset('dependable.s.04'),
 Synset('good.s.12'),
 Synset('good.s.13'),
 Synset('effective.s.04'),
 Synset('good.s.15'),
 Synset('good.s.16'),
 Synset('good.s.17'),
 Synset('good.s.18'),
 Synset('good.s.19'),
 Synset('good.s.20'),
 Synset('good.s.21'),
 Synset('well.r.01'),
 Synset('thoroughly.r.02')]
```

As shown here, we can see that traversing up the WordNet hierarchy will get us more general synsets as we move up. We started with 'house', in this case refering to the casino ("The house always wins!") and as we traversed up the hierarchy we can to see synsets like management, from management we went to administration and so on. So WordNet tries to be as broad and inclusive as possible.

That synset of house isn't great for hyponyms, meronyms, etc. So we'll try a different noun and see what comes up.

[ ]:
```
new_noun = wn.synset('good.n.02')

print("Hyponyms of Good: ", )
hyponyms = new_noun.hyponyms()
for hyponym in hyponyms:
    print(hyponym.name())
print("\n")
```

```
print("Meronyms of Good: ", )
meronyms = new_noun.part_meronyms()
for meronym in meronyms:
    print(meronym.name())
print("\n")

print("Holonyms of Good: ", )
holonyms = new_noun.part_holonyms()
for holonym in holonyms:
    print(holonym.name())
print("\n")

print("Antonyms of Good: ", )
#lemmatize
lemmas = new_noun.lemmas()
for lemma in lemmas:
    antonyms = lemma.antonyms()
    if antonyms:
        for antonym in antonyms:
            print(antonym.name())
print("\n")
```

```
Hyponyms of Good:
beneficence.n.02
benignity.n.01
kindness.n.01
saintliness.n.01
summum_bonum.n.01
virtue.n.01
virtue.n.04


Meronyms of Good:


Holonyms of Good:


Antonyms of Good:
evil
evilness
```

[ ]:
```
#selecting a verb
verb = wn.synsets('eat')
```

```
print(verb)
```

```
[Synset('eat.v.01'), Synset('eat.v.02'), Synset('feed.v.06'),
Synset('eat.v.04'), Synset('consume.v.05'), Synset('corrode.v.01')]
```

[ ]:
```
#extracting everything from verb = CONSUME
verb = wn.synset('consume.v.05')
print(verb)
print(verb.definition(),
print(verb.lemmas()),
print(verb.examples()))
print("\n")

#traversing up the hierarchy
print("Hypernyms of Consume: ", )
hypernyms = verb.hypernyms()
while hypernyms:
    print(hypernyms[0].name())
    hypernyms = hypernyms[0].hypernyms()
```

```
Synset('consume.v.05')
[Lemma('consume.v.05.consume'), Lemma('consume.v.05.eat_up'),
Lemma('consume.v.05.use_up'), Lemma('consume.v.05.eat'),
Lemma('consume.v.05.deplete'), Lemma('consume.v.05.exhaust'),
Lemma('consume.v.05.run_through'), Lemma('consume.v.05.wipe_out')]
['this car consumes a lot of gas', 'We exhausted our savings', 'They run through
20 bottles of wine a week']
use up (resources or materials) None None


Hypernyms of Consume:
spend.v.02
pay.v.01
give.v.03
transfer.v.05
```

WordNet's hierarchy of verbs is similar from how it organizes nouns, as it looks for common verbs while looking to get more broad and general. At the top of the WordNet verb hierarchy, we have really broad categories and the goal is to get closer to those broad categories as we traverse the hierarchy.

[ ]:
```
#morphy
print(wn.morphy("consume", wn.ADJ))
print(wn.morphy("consume", wn.VERB))
print(wn.morphy("consume", wn.NOUN))
```

```
None
consume
None
```

Staying on topic here, we'll evaluate how similar the words 'consume' and 'eat' are to each other.

```
[ ]: eat = wn.synset('eat.v.01')
consume = wn.synset('consume.v.01') #different defn: meaning eat immoderately

print(wn.wup_similarity(eat, consume))

#lesk
from nltk.wsd import lesk

sent = ['I', 'will', 'eat', 'grass']
sent2 = ['Thier', 'cheif', 'food', 'is', 'grass', 'but', 'they', 'also',␣
 ↪'consume', 'large', 'amounts', 'of', 'LOUD', 'ROOMMATES.']
print(lesk(sent, 'eat'))
print(lesk(sent2, 'consume'))

eatv2 = wn.synset('eat.v.02')
consumev2 = wn.synset('consume.v.02')

print(eatv2.definition())
print(consumev2.definition())
```

```
0.8571428571428571
Synset('eat.v.02')
Synset('consume.v.02')
eat a meal; take a meal
serve oneself to, or consume regularly
```

We see that the Wu-Palmer rated 'eat' and 'consume' as very similar, which is what I expected. Though the lesk algorithm detected different versions of the original verbs, which I didn't expect as much. Maybe my sample sentences aren't great to use here, as eat/consume can become pretty ambiguous depending on what you're talking about.

Sentiwordnet is used to assign WordNet synsets values that range from positive, negative and neutral. THis is typically referred to as sentiment analysis and has some interesting use cases. One potential use case I would think is particularly useful is scraping off of Twitter to find public opinon on certain companies to try and make a stock suggestion tool, however the depression detection in tweets example we discussed in class seems to be an infinitly more useful use case than anything I can think of, in my opinion.

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

rain = swn.senti_synset('rain.n.01')
print(rain)
print("Positive score = ", rain.pos_score())
print("Negative score = ", rain.neg_score())
print("Objective score = ", rain.obj_score())
```

```
sentence = 'Do not rain on my parade'
pos = 0
neg = 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('Positive:', pos)
print('Negative:', neg)
```

```
<rain.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
Positive: 0.0
Negative: 0.625
```

```
[nltk_data] Downloading package sentiwordnet to
[nltk_data]     /home/jacko/nltk_data…
[nltk_data]   Package sentiwordnet is already up-to-date!
```

We get a more negative score when we evalute the whole sentence, which I'm betting is exclusively because of the word "NOT" in the phrase "Do not rain on my parade", which makes sense, as rain isn't inherently bad or good when talking about the weather, I guess.

Collactions are when words appear together and it is determined that this is down to more than chance, meaning that those words share some kind of relationship.

[ ]:
```
import nltk
from nltk.book import text4

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
```

I want to focus on 'fellow citizens' just because it's the first one and said a lot in every day speech.

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

import math
```

```
vocab = len(set(text4))
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)
```

```
p(Fellow - Citizens):  0.0004987531172069825
p(Fellow):  0.002394014962593516
p(Citizens):  0.0006982543640897755
PMI:  8.220925288338247
```

Our PMI is really high! This means that we can conclude that "Fellow Citizens" is very likely to be a collocation.