wordnet

February 25, 2023

0.0.1 WordNet

WordNet is a large lexical database that contains many relations between words. It is akin to a thesaurus but for word relationships and figures of speech.

[1]:
```
NOUN = "friend"

from nltk.corpus import wordnet
synsets = wordnet.synsets(NOUN)
synsets
```

[1]:
```
[Synset('friend.n.01'),
 Synset('ally.n.02'),
 Synset('acquaintance.n.03'),
 Synset('supporter.n.01'),
 Synset('friend.n.05')]
```

[2]:
```
syn_noun = synsets[0]
print(syn_noun)
print(syn_noun.definition())
print(syn_noun.examples())
print(syn_noun.lemmas())

print("\traversal:")
while len(syn_noun := syn_noun.hypernyms()) != 0:
    syn_noun = syn_noun[0]
    print(syn_noun)
```

```
Synset('friend.n.01')
a person you know well and regard with affection and trust
['he was my best friend at the university']
[Lemma('friend.n.01.friend')]
        raversal:
Synset('person.n.01')
Synset('causal_agent.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

These seem to have some depth to them, while being very generalized all the way up. It removed the relationship of being a friend, then the human aspect, before being turned into the meta elements

of an entity.

[3]:
```
syn_noun = synsets[0]
print(f"hypernyms: {syn_noun.hypernyms()}")
print(f"hyponyms: {syn_noun.hyponyms()}")
print(f"meronyms: {syn_noun.member_meronyms()}")
print(f"holonyms: {syn_noun.member_holonyms()}")
print(f"antonym: {syn_noun.lemmas()[0].antonyms()}")
```

```
hypernyms: [Synset('person.n.01')]
hyponyms: [Synset('alter_ego.n.01'), Synset('amigo.n.01'),
Synset('best_friend.n.01'), Synset('brother.n.04'), Synset('buddy.n.01'),
Synset('companion.n.01'), Synset('confidant.n.01'), Synset('flatmate.n.01'),
Synset('girlfriend.n.01'), Synset('light.n.08'), Synset('mate.n.08'),
Synset('roommate.n.01'), Synset('schoolfriend.n.01')]
meronyms: []
holonyms: []
antonym: []
```

[4]:
```
VERB = "throw"

synsets = wordnet.synsets(VERB)
synsets
```

[4]:
```
[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')]
```

[5]:
```
syn_verb = synsets[5]
print(syn_verb)
print(syn_verb.definition())
```

```
print(syn_verb.examples())
print(syn_verb.lemmas())

print("\ntraversal:")
while len(syn_verb := syn_verb.hypernyms()) != 0:
    syn_verb = syn_verb[0]
    print(syn_verb)
```

```
Synset('throw.v.01')
propel through the air
['throw a frisbee']
[Lemma('throw.v.01.throw')]

traversal:
Synset('propel.v.01')
Synset('move.v.02')
```

The traversal seems to lead back to a general "move" verb. Once again, generalization happens, where the entity is removed, then the force is also removed. There seems to be a general shortening, as checking the other words in the list, the traversal length seems to be smaller.

[6]:
```
[
wordnet.morphy("throw", wordnet.VERB),
wordnet.morphy("throwing", wordnet.VERB),
wordnet.morphy("threw", wordnet.VERB),
wordnet.morphy("thrown", wordnet.VERB),
wordnet.morphy("throws", wordnet.VERB),
]
```

[6]:
```
['throw', 'throw', 'throw', 'throw', 'throw']
```

[7]:
```
# I like colors better
SIM1, SIM2 = "maroon", "rose"
for x in wordnet.synsets(SIM1):
    print(f"{x}: {x.definition()}")
print()
for x in wordnet.synsets(SIM2):
    print(f"{x}: {x.definition()}")
```

```
Synset('maroon.n.01'): a person who is stranded (as on an island)
Synset('maroon.n.02'): a dark purplish-red to dark brownish-red color
Synset('maroon.n.03'): an exploding firework used as a warning signal
Synset('maroon.v.01'): leave stranded or isolated with little hope of rescue
Synset('maroon.v.02'): leave stranded on a desert island without resources
Synset('maroon.s.01'): of dark brownish to purplish red

Synset('rose.n.01'): any of many shrubs of the genus Rosa that bear roses
Synset('blush_wine.n.01'): pinkish table wine from red grapes whose skins were
removed after fermentation began
```

```
Synset('rose.n.03'): a dusty pink color
Synset('rise.v.01'): move upward
Synset('rise.v.02'): increase in value or to a higher point
Synset('arise.v.03'): rise to one's feet
Synset('rise.v.04'): rise up
Synset('surface.v.01'): come to the surface
Synset('originate.v.01'): come into existence; take on form or shape
Synset('ascend.v.08'): move to a better position in life or to a better job;
"She ascended from a life of poverty to one of great
Synset('wax.v.02'): go up or advance
Synset('heighten.v.01'): become more extreme
Synset('get_up.v.02'): get up and out of bed
Synset('rise.v.11'): rise in rank or status
Synset('rise.v.12'): become heartened or elated
Synset('rise.v.13'): exert oneself to meet a challenge
Synset('rebel.v.01'): take part in a rebellion; renounce a former allegiance
Synset('rise.v.15'): increase in volume
Synset('rise.v.16'): come up, of celestial bodies
Synset('resurrect.v.03'): return from the dead
Synset('rose.s.01'): of something having a dusty purplish pink color
```

[8]:
```
syn_sim1, syn_sim2 = wordnet.synsets(SIM1)[1], wordnet.synsets(SIM2)[2]
print(f"Wu-Palmer similarity: {wordnet.wup_similarity(syn_sim1, syn_sim2)}")

from nltk.wsd import lesk
sentence = "The maroon blood stained the rose blanket which the alien slept on.
 ↪".split()
print(f"maroon lesk: {lesk(sentence, 'maroon', 'n')}")
print(f"rose lesk: {lesk(sentence, 'rose', 'n')}")
```

```
Wu-Palmer similarity: 0.7368421052631579
maroon lesk: Synset('maroon.n.03')
rose lesk: Synset('rose.n.01')
```

While Wu-Palmer seems to be fairly correct (maroon and rose are both colors, a subset of red), lesk seems to be woefully inadequite in finding the definition of the words. It chose a definition that was fundamentally wrong and obscure for maroon, and misinterpreted the definition of rose (possibly thinking of a rose "bed").

For SentiWordNet, it's usefulness could be only deligated to determining general consensous about some opinion, possibly with determining some level of sarcasm and "joke-ness" of a statement. This could be expanded to two things: sentiment analysis of social media posting as a sort of semi-automated flag system, and chatbot frustration analysis (though, I can imagine that when it gets the analysis wrong it won't just anger the person more).

[9]:
```
EMOTE = "communist"

for x in wordnet.synsets(EMOTE):
    print(f"{x}: {x.definition()}")
```

```
EMOTE = "communist.a.01"
print()
from nltk.corpus import sentiwordnet
bd = sentiwordnet.senti_synset(EMOTE)
print(f"Positivity: {bd.pos_score()}")
print(f"Negativity: {bd.neg_score()}")
print(f"Objectivity: {bd.obj_score()}")
```

```
Synset('communist.n.01'): a member of the communist party
Synset('communist.n.02'): a socialist who advocates communism
Synset('communist.a.01'): relating to or marked by communism

Positivity: 0.0
Negativity: 0.125
Objectivity: 0.875
```

[10]:
```
sentence = "The damn realists are not helping people that are getting␣
 ↪flagellated in the streets.".split()
for word in sentence:
    syns = list(sentiwordnet.senti_synsets(word))
    print(f"{word}: ", end="")
    if syns:
        print(f"(pos: {syns[0].pos_score():0.3f}, neg: {syns[0].neg_score():0.
 ↪3f})")
    else:
        print("no score")
```

```
The: no score
damn: (pos: 0.125, neg: 0.125)
realists: (pos: 0.000, neg: 0.000)
are: (pos: 0.000, neg: 0.000)
not: (pos: 0.000, neg: 0.625)
helping: (pos: 0.000, neg: 0.000)
people: (pos: 0.000, neg: 0.000)
that: no score
are: (pos: 0.000, neg: 0.000)
getting: (pos: 0.000, neg: 0.000)
flagellated: (pos: 0.000, neg: 0.000)
in: (pos: 0.000, neg: 0.000)
the: no score
streets.: no score
```

The majority of the words here seem to not actually get assigned a polarity score, despite this being an obviously negative sentence. The only words that seem to effect the score at all are “damn” and “not.” “Damn” makes some sense (eg “those damn blues” or “damn you to hell”) but “not” is a general negation word. The utility doesn’t seem all that great if the general system doesn’t seem to cover all words (eg “flagellated” is very easily negative in almost , if not all, contexts).

Collocations are groups of words that refer to a special object that cannot be substituted with other words. An example of this is the term "overhead light", where "overhead" cannot be swapped out with any other words that are synonyms of each, because it would change the definition of the object. (You could use "hanging", but that would make a term that is a hyponym.)

[11]:
```
from nltk.book import text4
from math import log2

text4.collocations()
text = ' '.join(text4.tokens)
vocab = len(set(text))

COLLOC = "one another"
print(f"PMI SCORE: {log2((text.count(COLLOC)/vocab)/((text.count('one')/
  ↪vocab)*(text.count('another')/vocab)))}")
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, …, text9 and sent1, …, sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
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
PMI SCORE: -4.4479598258014175
```

The PMI score seems to indicate that this isn't a collocation, but just a pattern. NLTK might have found something else that PMI might have missed. My interpretation of the PMI is just a one line generic implementation.