Jonathan Ho

CS 4395

WordNet Exploration

What is WordNet?

WordNet is a whole network of words connected lexically and semantically. It is almost akin to a dictionary for computers to interpret English words. Different types of words are grouped into synsets (words that express the same concept), and are interlinked.

Synset of a noun

```
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn
from nltk.book import text4
import math


# Get synsets of the noun "time"

wn.synsets('time')

     [Synset('time.n.01'),
      Synset('time.n.02'),
      Synset('time.n.03'),
      Synset('time.n.04'),
      Synset('time.n.05'),
      Synset('time.n.06'),
      Synset('clock_time.n.01'),
      Synset('fourth_dimension.n.01'),
      Synset('meter.n.04'),
      Synset('prison_term.n.01'),
      Synset('clock.v.01'),
      Synset('time.v.02'),
      Synset('time.v.03'),
      Synset('time.v.04'),
      Synset('time.v.05')]


# Definition of "time"

wn.synset('time.n.04').definition()

     'a suitable moment'


# Usage of "time"

wn.synset('time.n.04').examples()

     ['it is time to go']


# Lemmas of "time"

wn.synset('time.n.01').lemmas()

     [Lemma('time.n.01.time'), Lemma('time.n.01.clip')]


# Traversing up the hiearchy for "time"

hyp = lambda x: x.hypernyms()
list(wn.synset('time.n.04').closure(hyp))

     [Synset('moment.n.01'),
      Synset('point.n.06'),
      Synset('measure.n.02'),
      Synset('abstraction.n.06'),
      Synset('entity.n.01')]
```

It seems that the hiearchy of a noun in WordNet is organized on concept. Each higher level above the noun is a broader concept of the noun. For example moment could be a concept of time, which is one concept of a point, which is then one concept of measure. This goes on until it gets to the very top which is the word entity since it is the base concept of all nouns.

```
# Hypernyms, hyponyms, meronyms, holonyms, and antonyms of "time"

time = wn.synset('time.n.04')

print('Hypernyms: ', time.hypernyms())
print('Hyponyms: ', time.hyponyms())
print('Meronyms: ', time.part_meronyms())
print('Holonyms: ', time.part_holonyms())
print('Antonym: ', time.lemmas()[0].antonyms())
```

```
Hypernyms:  [Synset('moment.n.01')]
Hyponyms:  [Synset('high_time.n.01'), Synset('occasion.n.04')]
Meronyms:  []
Holonyms:  []
Antonym:  []
```

Synset of a verb

```
# Get synsets of the verb "move"

wn.synsets('move')
```

```
[Synset('move.n.01'),
 Synset('move.n.02'),
 Synset('motion.n.03'),
 Synset('motion.n.06'),
 Synset('move.n.05'),
 Synset('travel.v.01'),
 Synset('move.v.02'),
 Synset('move.v.03'),
 Synset('move.v.04'),
 Synset('go.v.02'),
 Synset('be_active.v.01'),
 Synset('move.v.07'),
 Synset('act.v.01'),
 Synset('affect.v.05'),
 Synset('motivate.v.01'),
 Synset('move.v.11'),
 Synset('move.v.12'),
 Synset('move.v.13'),
 Synset('move.v.14'),
 Synset('move.v.15'),
 Synset('move.v.16')]
```

```
# Definition, example, and lemmas of "act"

act = wn.synsets('move')[12]

print('Definition: ', act.definition())
print('Examples: ', act.examples())
print('Lemmas: ', act.lemmas())
```

```
Definition:  perform an action, or work out or perform (an action)
Examples:  ['think before you act', 'We must move quickly', 'The governor should act on the new energy bill', 'The nanny acted quickly
Lemmas:  [Lemma('act.v.01.act'), Lemma('act.v.01.move')]
```

```
# Traverse up the hiearchy for "act"

hyp = lambda y: y.hypernyms()
list(act.closure(hyp))
```

```
[]
```

Unlike nouns, it seems it is more difficut to conceptualize a verb. For the word "act", it was not able to find any word higher up. Most likely because there is no underlying concept for the word "act".

```
# Using morphy to find different forms of "act"

print("Adjectives: ", wn.morphy('act', wn.ADJ))
print("Verbs: ", wn.morphy('act', wn.VERB))
print("Noun: ", wn.morphy('act', wn.NOUN))
```

```
Adjectives:  None
Verbs:  act
Noun:  act
```

Word Similarity Check

```
# List of synsets for "Fish"

wn.synsets('fish')
```

```
[Synset('fish.n.01'),
 Synset('fish.n.02'),
 Synset('pisces.n.02'),
 Synset('pisces.n.01'),
 Synset('fish.v.01'),
 Synset('fish.v.02')]
```

```
# List of synsets for "Sushi"

wn.synsets('sushi')
```

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

```
# Wu-Palmer similarity metric

fish = wn.synsets('fish')[0]
sushi = wn.synsets('sushi')[0]

wn.wup_similarity(fish, sushi)
```

```
0.21052631578947367
```

```
# Lesk algorithm for "fish" and "sushi"

sent = ['I', 'had', 'sushi', 'for', 'dinner', 'which', 'is', 'fish', 'wrapped', 'in', 'bite', 'sized', 'rolls', 'of', 'rice', '.']
print(lesk(sent, 'fish', 'n'))
print(lesk(sent,'sushi', 'n'))
```

```
Synset('pisces.n.01')
Synset('sushi.n.01')
```

```
# Definition of "pisces"

print(wn.synsets('fish')[3].definition())
```

```
the twelfth sign of the zodiac; the sun is in this sign from about February 19 to March 20
```

Looking at the Wu-Palmer similarity, the similarity score came out to be about 21%. I thought it would be a bit higher since sushi is essentially fish, but the concept of sushi might be too specific to the computer. As for the Lesk algorithm, while "sushi" could only has one synset to pick from, "fish" had chosen the 4th synset of its list. Even though the definition does not fit, it most likely picked that since it probably matched words in my sentence such as "in" and "is".

SentiWordNet

SentiWordNet is a database that scores how positive or negative a word is based off of opinion. Since it is how people perceive a word, it could be used to write a heartfelt message finding words that are seen as more "positive".

```
# Finding senti-synsets of the word "wonderful"

w = swn.senti_synset('wonderful.a.01')
```

```
# Printing out polarity scores
print(w)
print("Positive score = ", w.pos_score())
print("Negative score = ", w.neg_score())
print("Objective score = ", w.obj_score())
```

```
     <fantastic.s.02: PosScore=0.75 NegScore=0.0>
     Positive score =  0.75
     Negative score =  0.0
     Objective score =  0.25
```

```
# Printing polarity of each word in a sentence

sent = 'today i wanted to eat a croissant'
tokens = sent.split()

for token in tokens:
  syn_list = list(swn.senti_synsets(token))
  if syn_list:
    syn = syn_list[0]
    print(syn)
    print("Positive score = ", syn.pos_score())
    print("Negative score = ", syn.neg_score())
    print("Objective score = ", syn.obj_score())
    print()
```

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

     <iodine.n.01: PosScore=0.0 NegScore=0.0>
     Positive score =  0.0
     Negative score =  0.0
     Objective score =  1.0

     <desire.v.01: PosScore=0.25 NegScore=0.0>
     Positive score =  0.25
     Negative score =  0.0
     Objective score =  0.75

     <eat.v.01: PosScore=0.0 NegScore=0.0>
     Positive score =  0.0
     Negative score =  0.0
     Objective score =  1.0

     <angstrom.n.01: PosScore=0.0 NegScore=0.0>
     Positive score =  0.0
     Negative score =  0.0
     Objective score =  1.0

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

Looking at the scores for the synset of the word "wonderful", it makes sense that it would score a high positive score. Since it is an adjective, it is used to describe something in a positive way. In the made up sentence, I expected it to be mostly all objective scores, but I was surprised to see that "today" and "wanted" had a bit into the positive scores. Perhaps these words are used a bit in positive contexts. In NLP, the scores could help the computer potentially understand if a sentence is giving an overall positive or negative connotation.

Collocation

A collocation is two or more words that seem to end up together more often, almost colloquially. These words make up a meaning that has a unique definition where no other words can be synonimous.

```
# Collocations for text4

print(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
None
```

```
# Calculating mutual information of "old world"

text = ' '.join(text4.tokens)
vocab = len(set(text4))
ow = text.count('Old World')/vocab
print("p(Old World) = ", ow)
o = text.count('Old')/vocab
print("p(Old) = ", o)
w = text.count('World')/vocab
print("p(World) = ", w)
pmi = math.log2(ow / (o * w))
print("pmi = ", pmi)
```

```
p(Old World) =  0.000997506234413965
p(Old) =  0.0010972568578553616
p(World) =  0.0017955112219451373
pmi =  8.983886091037398
```

The mutual information of the words "Old World" resulted in a positive value. This means that it is likely a collocation. If there were values to compare with and this value ended up being higher than the comparison values, then it would imply it is even more likely that "Old World" is a collocation than other phrases. If the value had ended up being 0 or lower, then the words would not have been a collocation.

✓ 0s completed at 5:36 PM