Homework 3: WordNet

CS 4395.002 Human Language Technologies

Dr. Mazidi

Henry Kim HTK180000

1. WordNet Summary

WordNet is database of English words maintained by Princeton University that organizes each word into synsets based on its different defintions. It can be used to find words that have a similar meaning, words of the same meaning that are more specific, or words of the same meaning that are less specific.

Princeton University "About WordNet." WordNet. Princeton University. 2010.

```
# Import libraries
import math
import nltk
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn
from nltk.sentiment.vader import SentimentIntensityAnalyzer
# Download nltk data
nltk.download('all')
from nltk.book import *
```

2. Noun Synsets

Here are the synsets of the word 'light'

```
for synset in wn.synsets('light'):
  print(synset)
  print('\t' + synset.name())
  print('\t' + synset.definition())
```

```
    Synset('light.n.01')
            light.n.01
            (physics) electromagnetic radiation that can produce a visual sensation
    Synset('light.n.02')
            light.n.02
            any device serving as a source of illumination
    Synset('light.n.03')
            light.n.03
```

✓ 0s completed at 8:12 PM

```
Synset('luminosity.n.01')
        luminosity.n.01
        the quality of being luminous; emitting or reflecting light
Synset('light.n.05')
        light.n.05
        an illuminated area
Synset('light.n.06')
        light.n.06
        a condition of spiritual awareness; divine illumination
Synset('light.n.07')
        light.n.07
        the visual effect of illumination on objects or scenes as created in picture
Synset('light.n.08')
        light.n.08
        a person regarded very fondly
Synset('light.n.09')
        light.n.09
        having abundant light or illumination
Synset('light.n.10')
        light.n.10
        mental understanding as an enlightening experience
Synset('sparkle.n.01')
        sparkle.n.01
        merriment expressed by a brightness or gleam or animation of countenance
Synset('light.n.12')
        light.n.12
        public awareness
Synset('inner_light.n.01')
        inner_light.n.01
        a divine presence believed by Quakers to enlighten and guide the soul
Synset('light.n.14')
        light.n.14
        a visual warning signal
Synset('lighter.n.02')
        lighter.n.02
        a device for lighting or igniting fuel or charges or fires
Synset('light.v.01')
        light.v.01
        make lighter or brighter
Synset('light_up.v.05')
        light_up.v.05
        begin to smoke
Synset('alight.v.01')
        alight.v.01
        to come to rest, settle
Synset('ignite.v.01')
        ignite.v.01
        cause to start burning; subject to fire or great heat
Synset('fall.v.20')
```

3. Noun Definition, Usage, and Lemmas

Let's look at the first synset definition and usage

```
noun = wn.synset('light.n.01')
print("Definition:")
print('\t' + noun.definition())

print("Usages:")
for example in noun.examples():
  print('\t' + example)

print("Lemmas:")
for lemma in noun.lemmas():
  print('\t' + str(lemma))
```

```
    Definition:
            (physics) electromagnetic radiation that can produce a visual sensation
    Usages:
            the light was filtered through a soft glass window
    Lemmas:
            Lemma('light.n.01.light')
            Lemma('light.n.01.visible_light')
            Lemma('light.n.01.visible_radiation')
```

Now we traverse up the hierarchy of hypernyms of 'light' according to its first definition. A hypernym of a word is a more abstract or general version of it. The hierarchy goes from more specific nouns at the bottom to more general at the top of the hypernyms, and it ends with the broad category an 'entity' that may or may not even exist at the root.

```
hyp = noun.hypernyms()[0]
top = noun.root_hypernyms()

while hyp:
    print(hyp)
    print('\t' + hyp.name())
    print('\t' + hyp.definition())
    if hyp in top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
    Synset('actinic_radiation.n.01')
            actinic_radiation.n.01
            electromagnetic radiation that can produce photochemical reactions
    Synset('electromagnetic_radiation.n.01')
            electromagnetic_radiation.n.01
            radiation consisting of waves of energy associated with electric and magneti
    Synset('radiation.n.01')
            radiation.n.01
            energy that is radiated or transmitted in the form of rays or waves or parti
```

```
Synset('energy.n.01')
        energy.n.01
        (physics) a thermodynamic quantity equivalent to the capacity of a physical
Synset('physical_phenomenon.n.01')
        physical_phenomenon.n.01
        a natural phenomenon involving the physical properties of matter and energy
Synset('natural_phenomenon.n.01')
        natural_phenomenon.n.01
        all phenomena that are not artificial
Synset('phenomenon.n.01')
        phenomenon.n.01
        any state or process known through the senses rather than by intuition or re
Synset('process.n.06')
        process.n.06
        a sustained phenomenon or one marked by gradual changes through a series of
Synset('physical_entity.n.01')
        physical_entity.n.01
        an entity that has physical existence
Synset('entity.n.01')
        entity.n.01
        that which is perceived or known or inferred to have its own distinct existe
```

4. Hypernyms, Hyponyms, Meronyms, Holonyms, and Antonyms

Along with hypernyms, WordNet also includes hyponyms (word that is more specific than the base word), meronyms (word that makes up part of the base word), holonnyms (word that base word makes up a part of), and antonyms (word with opposite meaning of base word) for some words.

```
print("Hypernyms:")
for hyp in noun.hypernyms():
  print('\t' + str(hyp))

print("Hyponyms:")
for hyp in noun.hyponyms():
  print('\t' + str(hyp))

print("Part Meronyms:")
for mer in noun.part_meronyms():
  print('\t' + str(mer))

print("Substance Meronyms:")
for mer in noun.substance_meronyms():
  print('\t' + str(mer))

print("Part Holonyms:")
for mer in noun.part_holonyms():
  print('\t' + str(mer))
```

```
print("Substance Holonyms:")
for mer in noun.substance_holonyms():
  print('\t' + str(mer))

antonyms = []
for lemma in noun.lemmas():
  antonyms += lemma.antonyms()

print("Antonyms:")
for ant in antonyms:
  print('\t' + ant)
```

```
    Hypernyms:
            Synset('actinic_radiation.n.01')
    Hyponyms:
            Synset('beam.n.04')
            Synset('candlelight.n.01')
            Synset('corona.n.04')
            Synset('counterglow.n.01')
            Synset('daylight.n.02')
            Synset('firelight.n.01')
            Synset('fluorescence.n.01')
            Synset('friar's_lantern.n.01')
            Synset('gaslight.n.01')
            Synset('glow.n.05')
            Synset('half-light.n.01')
            Synset('incandescence.n.01')
            Synset('lamplight.n.01')
            Synset('luminescence.n.01')
            Synset('meteor.n.02')
            Synset('moonlight.n.01')
            Synset('radiance.n.01')
            Synset('scintillation.n.01')
            Synset('starlight.n.01')
            Synset('streamer.n.01')
            Synset('sunlight.n.01')
            Synset('torchlight.n.01')
            Synset('twilight.n.02')
    Part Meronyms:
    Substance Meronyms:
    Part Holonyms:
            Synset('electromagnetic_spectrum.n.01')
    Substance Holonyms:
    Antonyms:
```

5. Verb Synsets

Here are the synsets for the word 'sail.'

```
for synset in wn.synsets('sail'):
  print(synset)
```

```
  print('\t' + synset.name())
  print('\t' + synset.definition())
```

```
    Synset('sail.n.01')
            sail.n.01
            a large piece of fabric (usually canvas fabric) by means of which wind is us
    Synset('cruise.n.01')
            cruise.n.01
            an ocean trip taken for pleasure
    Synset('sail.n.03')
            sail.n.03
            any structure that resembles a sail
    Synset('sail.v.01')
            sail.v.01
            traverse or travel on (a body of water)
    Synset('sweep.v.02')
            sweep.v.02
            move with sweeping, effortless, gliding motions
    Synset('sail.v.03')
            sail.v.03
            travel on water propelled by wind
    Synset('voyage.v.01')
            voyage.v.01
            travel on water propelled by wind or by other means
```

6. Verb Definition, Usage, and Lemmas

```
verb = wn.synset('sail.v.01')
print("Definition:")
print('\t' + verb.definition())

print("Usages:")
for example in verb.examples():
  print('\t' + example)

print("Lemmas:")
for lemma in verb.lemmas():
  print('\t' + str(lemma))
```

```
    Definition:
            traverse or travel on (a body of water)
    Usages:
            We sailed the Atlantic
            He sailed the Pacific all alone
    Lemmas:
            Lemma('sail.v.01.sail')
```

Now we traverse up the hierarchy of hypernyms of 'sail' according to its first verb definition. Like with nouns, the hierarchy of verbs starts more specific at the bottom and and more general at

the top. Here the verb gets more general in 'travel' to mean traveling in general as opposed to just travel be sea. It then gets even more general to include traveling even in a metaphorical sense.

```
hyp = verb.hypernyms()[0]
top = verb.root_hypernyms()

while hyp:
    print(hyp)
    print('\t' + hyp.name())
    print('\t' + hyp.definition())
    if hyp in top:
        break
    if hyp.hypernyms():
        hyp = hyp.hypernyms()[0]
```

```
 Synset('travel.v.04')
        travel.v.04
        travel upon or across
 Synset('travel.v.01')
        travel.v.01
        change location; move, travel, or proceed, also metaphorically
```

7. Different Forms

Here are different forms of the word 'sail' according to morphy. Since it is rules-based, its capabilities are limited.

```
print(wn.morphy('sailor'))
print(wn.morphy('sailor'), wn.ADJ)
print(wn.morphy('sailor'), wn.VERB)
print(wn.morphy('sailor'), wn.NOUN)
print(wn.morphy('sailing'))
print(wn.morphy('sailing'), wn.ADJ)
print(wn.morphy('sailing'), wn.VERB)
print(wn.morphy('sailing'), wn.NOUN)
print(wn.morphy('sailed'))
print(wn.morphy('sailed'), wn.ADJ)
print(wn.morphy('sailed'), wn.VERB)
print(wn.morphy('sailed'), wn.NOUN)
```

```
 sailor
 sailor a
 sailor v
 sailor n
 sailing
 sailing a
```

```
sailing v
sailing n
sail
sail a
sail v
sail n
```

8. Word Similarity

Let's compare the similarity of words 'lunch' and 'sandwich.' These words can both be found in the context of having a meal, but neither metric rates them as very similar.

```
# Use lesk to pick the definitions where the context of the words overlap.
context = ['lunch', 'sandwich']
word1 = lesk(context, 'lunch')
word2 = lesk(context, 'sandwich')

print('Path similarity', wn.path_similarity(word1, word2))
print('Wu-Palmer similarity', wn.wup_similarity(word1, word2))
```

```
Path similarity 0.09090909090909091
Wu-Palmer similarity 0.16666666666666666
```

The synonyms 'carry' and 'hold' are more similar according to both metrics, but Wu-Palmer rates them as more similar.

```
# Use lesk to pick the definitions where the context of the words overlap.
context = ['carry', 'hold']
word1 = lesk(context, 'carry')
word2 = lesk(context, 'hold')

print('Path similarity', wn.path_similarity(word1, word2))
print('Wu-Palmer similarity', wn.wup_similarity(word1, word2))
```

```
Path similarity 0.2
Wu-Palmer similarity 0.3333333333333333
```

Although 'high' and 'low' are antonyms, they are rated as similar likely due to both describing altitude.

```
# Use lesk to pick the definitions where the context of the words overlap.
context = ['high', 'low']
word1 = lesk(context, 'high')
word2 = lesk(context, 'low')
```

```
print('Path similarity', wn.path_similarity(word1, word2))
print('Wu-Palmer similarity', wn.wup_similarity(word1, word2))
```

```
Path similarity 0.25
Wu-Palmer similarity 0.4
```

9. SentiWordNet

SentiWordNet enhances WordNet by providing data about emotional connotation for each synset in the form of positive, negative, and objective scores. These can be used for evaluating the emotional content of text such as determining if an article is written to evoke a strongly negative reaction in readers to encourage its spread rather than providing objective truth. For example, the word 'fear' has mostly negative emotional connotations and implies there could be danger that requires action.

```
senti_list = list(swn.senti_synsets('fear'))
for item in senti_list:
    print(item)
    print('\tPositive score:', item.pos_score())
    print('\tNegative score:', item.neg_score())
    print('\tObjective score:', item.obj_score())
```

```
 <fear.n.01: PosScore=0.0 NegScore=0.875>
        Positive score: 0.0
        Negative score: 0.875
        Objective score: 0.125
 <concern.n.02: PosScore=0.125 NegScore=0.75>
        Positive score: 0.125
        Negative score: 0.75
        Objective score: 0.125
 <fear.n.03: PosScore=0.5 NegScore=0.0>
        Positive score: 0.5
        Negative score: 0.0
        Objective score: 0.5
 <fear.v.01: PosScore=0.125 NegScore=0.625>
        Positive score: 0.125
        Negative score: 0.625
        Objective score: 0.25
 <fear.v.02: PosScore=0.0 NegScore=0.625>
        Positive score: 0.0
        Negative score: 0.625
        Objective score: 0.375
 <fear.v.03: PosScore=0.0 NegScore=0.5>
        Positive score: 0.0
        Negative score: 0.5
        Objective score: 0.5
 <fear.v.04: PosScore=0.0 NegScore=0.625>
        Positive score: 0.0
        Negative score: 0.625
```

```
        Objective score: 0.375
<reverence.v.01: PosScore=0.625 NegScore=0.0>
        Positive score: 0.625
        Negative score: 0.0
        Objective score: 0.375
```

When analyzing the sentence 'that smoothie was delicious,' the SentiWordNet gives it an overall positive score. I was surprised that the postivity comes from 'smoothie' and not 'delicious.' Theoretically, this tool could be useful for determining if restaurant reviews were favorable, but care would have to be taken to ensure there is not bias about which product is being reviewed, such as a 'smoothie' being an inherently positive word even if the customer did not actually enjoy it.

```
sentence = 'That smoothie was delicious.'
tokens = nltk.word_tokenize(sentence)
pos = 0
neg = 0
for token in tokens:
    print(token)
    senti_list = list(swn.senti_synsets(token))
    if senti_list:
        syn = senti_list[0]
        print(syn)
        neg += syn.neg_score()
        pos += syn.pos_score()
print('Positive total:', pos)
print('Negative total:', neg)
```

```
That
smoothie
<smoothie.n.01: PosScore=0.625 NegScore=0.0>
was
<washington.n.02: PosScore=0.0 NegScore=0.0>
delicious
<delicious.n.01: PosScore=0.0 NegScore=0.0>
.
Positive total: 0.625
Negative total: 0.0
```

For comparison, here is an analysis by VADER, which also gives the sentence a postive score.

```
analyzer = SentimentIntensityAnalyzer()
vs = analyzer.polarity_scores(sentence)
print(sentence, '\n\t', str(vs))
```

```
That smoothie was delicious.
         {'neg': 0.0, 'neu': 0.448, 'pos': 0.552, 'compound': 0.5719}
```

10. Collocation

A collocation occurs when multiple words appear together to form a new meaning that is different from what combining their standard definitions would yield. For example, a 'paper weight' refers to a small object that keeps paper on a desk and not a weight made of paper. Here are the collocations found in the Inaugural Address corpus of nltk.

```
for colloc in text4.collocation_list():
  print(colloc)
```

```
    ('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')
```

Now we calculate the mutual information of the collocation 'Vice President' using the formula P(x,y)=[P(x)*P(y)]. We can also calculate the mutual information of 'every' and 'President' to get a baseline comparison. The collocation 'Vice President' has a relatively high mutual information score, which indicates that Vice President occurs more often than random chance would dictate. 'Vice President' is a significant collocation that refers to a specific political position that 'President' and any other adjective would not describe.

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

vp = text.count('Vice President') / vocab
print("p(Vice President) = ", vp)
v = text.count('Vice') / vocab
print("p(Vice) = ", v)
```

```
p = text.count('President') / vocab
print('p(President) = ', p)
pmi = math.log2(vp / (v * p))
print('pmi = ', pmi)
print()

ep = text.count('every President') / vocab
print("p(every President) = ", ep)
e = text.count('every') / vocab
print("p(every) = ", e)
print('p(President) = ', p)
pmi = math.log2(ep / (e * p))
print('pmi = ', pmi)
```

```
p(Vice President) =  0.0017955112219451373
p(Vice) =  0.0018952618453865336
p(President) =  0.010773067331670824
pmi =  6.458424602064904

p(every President) =  9.975062344139652e-05
p(every) =  0.03291770573566085
p(President) =  0.010773067331670824
pmi =  -1.8298951001796395
```