Wordnet Assignment

Liam Leece - lcl180002

In this assignment we study WordNet, which is a tool provided by nltk. Wordnet allows us to examine many different aspects of words such as synonyms, frequencies, and connotations. The code segment below shows the synonym set of the word chicken.

```
import nltk
nltk.download("wordnet")
nltk.download("omw-1.4")
from nltk.corpus import wordnet as wn
nltk.download('sentiwordnet')
from nltk.corpus import sentiwordnet as swn
wn.synsets('chicken')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[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 sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
[Synset('chicken.n.01'),
 Synset('chicken.n.02'),
 Synset('wimp.n.01'),
 Synset('chicken.n.04'),
 Synset('chicken.s.01')]
```

The segment below takes the noun from above and dives deeper into it, displaying the meaning, the lemmas, and examples if available. This loop also shows the heirarchy of the noun system in WordNet. Almost all nouns can be reached by using their noun tree and this shows how we can work our way up from a chicken to a vague entity.

```
print(wn.synset('chicken.n.01').definition())
print(wn.synset('chicken.n.01').lemmas())
print(wn.synset('chicken.n.01').examples())
hyp = wn.synset('chicken.n.01').hypernyms()[0]
top = wn.synset('entity.n.01')
while hyp:
  print(hyp)
  if hyp == top:
    break
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
the flesh of a chicken used for food
```

```
[Lemma('chicken.n.01.chicken'), Lemma('chicken.n.01.poulet'), Lemma('chicken.n.01.volail
[]
Synset('poultry.n.02')
Synset('bird.n.02')
Synset('meat.n.01')
Synset('food.n.02')
Synset('solid.n.01')
Synset('matter.n.03')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

This segment shows the different aspects of our chosen noun, including things like parts of a whole and what a chicken can come from.

```
print(wn.synset('chicken.n.01').hypernyms())
print(wn.synset('chicken.n.01').hyponyms())
print(wn.synset('chicken.n.01').part_meronyms())
print(wn.synset('chicken.n.01').part_holonyms())
antonyms = []

for syn in wn.synsets("chicken"):
    for i in syn.lemmas():
         if i.antonyms():
              antonyms.append(i.antonyms()[0].name())

print(set(antonyms))
```

```
[Synset('poultry.n.02')]
[Synset('broiler.n.02'), Synset('capon.n.01'), Synset('fryer.n.01'), Synset('hen.n.03'),
[Synset('breast.n.03'), Synset('chicken_wing.n.01')]
[Synset('chicken.n.02')]
set()
```

```
print(wn.synsets("breathe"))
```

```
[Synset('breathe.v.01'), Synset('breathe.v.02'), Synset('breathe.v.03'), Synset('breathe
```

This segment is very similar to our traversal of the noun tree. However the verb tree in WordNet is not nearly as extensive. This verb only had 5 hypernyms until it reached the top of its line.

```
br = wn.synset('rest.v.02')
print(br.definition())
print(br.lemmas())
print(br.examples())
```

```
hyp = br.hypernyms()[0]
for i in range(5):
  print(hyp)
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
take a short break from one's activities in order to relax
[Lemma('rest.v.02.rest'), Lemma('rest.v.02.breathe'), Lemma('rest.v.02.catch_one's_breat
[]
Synset('pause.v.02')
Synset('interrupt.v.01')
Synset('break.v.10')
Synset('end.v.02')
Synset('change.v.01')
```

```
print(wn.morphy('rest', wn.ADJ))
print(wn.morphy('rest', wn.ADV))
print(wn.morphy('rest', wn.NOUN))
```

```
None
None
rest
```

As I expected, the Wu Palmer similarity of the words river and creek is very high. The closer the number is to 1, the more similar the metric finds the words. The lesk algorithm then determines which use of the word river I intended by using word associations.

```
river = wn.synset('river.n.01')
creek = wn.synset('creek.n.01')
print(wn.wup_similarity(river, creek))
from nltk.wsd import lesk
sent = ['The', 'river', 'was', 'raging', '.']
print(lesk(sent, 'river', 'n'))
```

```
0.8333333333333334
Synset('river.n.01')
```

The next segment of code explores the positive and negative associations with words. The word I chose was crush because it can be to destroy something or to be infatuated, which have two very different associations. Overall the scor was objective but still a little positive. Knowing that its a positive association can help determine which sense of the word I was using.

```
crush = swn.senti_synset('crush.n.03')
print(crush)
print("Pos score = ", crush.pos_score())
print("Neg score = ", crush.neg_score())
```

```
print("Obj score = ", crush.obj_score())

sent = ['The', 'haunted', 'house', 'was', 'terribly', 'frightening', 'but', 'so', 'fun', '!']
for t in sent:
  neg = 0
  pos = 0
  syn_list = list(swn.senti_synsets(t))
  if syn_list:
    syn = syn_list[0]
    neg += syn.neg_score()
    pos += syn.pos_score()
    print(t, ":\tneg\tpos counts")
    print('\t', neg, '\t', pos)
```

```
 <puppy_love.n.01: PosScore=0.375 NegScore=0.0>
 Pos score =  0.375
 Neg score =  0.0
 Obj score =  0.625
 haunted :       neg     pos counts
          0.0     0.0
 house : neg     pos counts
          0.0     0.0
 was :   neg     pos counts
          0.0     0.0
 terribly :      neg     pos counts
          0.0     0.25
 frightening :   neg     pos counts
          0.125   0.25
 but :   neg     pos counts
          0.0     0.0
 so :    neg     pos counts
          0.0     0.0
 fun :   neg     pos counts
          0.0     0.375
```

The collocations tool is used to find words that are often found next to each other in a text. Below are the most common collocations in the inaugural address.

```
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('stopwords')
from nltk.book import text4
from nltk.collocations import BigramCollocationFinder, BigramAssocMeasures
print(text4.collocations())
bigram_measures = BigramAssocMeasures()
```

```
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
[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]   Package stopwords is already up-to-date!
```