WordNet Assignment

Alexis Jennings

aej190000

CS 4395.001

Summary of WordNet

WordNet is a hierarchically-organized dictionary specifically designed for NLP purposes. The database uses "synsets" (synonym sets) to look up words in WordNet. Synsets are connected to other synsets via a hierarchy of semantic relations.

```
import nltk
nltk.download('omw-1.4')
nltk.download("wordnet")
nltk.download('sentiwordnet')
nltk.download('book')
```

```
    [nltk_data]    | Downloading package state_union to /root/nltk_data...
    [nltk_data]    |   Package state_union is already up-to-date!
    [nltk_data]    | Downloading package stopwords to /root/nltk_data...
    [nltk_data]    |   Package stopwords is already up-to-date!
    [nltk_data]    | Downloading package swadesh to /root/nltk_data...
    [nltk_data]    |   Package swadesh is already up-to-date!
    [nltk_data]    | Downloading package timit to /root/nltk_data...
    [nltk_data]    |   Package timit 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 toolbox to /root/nltk_data...
    [nltk_data]    |   Package toolbox is already up-to-date!

    [nltk_data]    | Downloading package udhr to /root/nltk_data...
    [nltk_data]    |   Package udhr is already up-to-date!
    [nltk_data]    | Downloading package udhr2 to /root/nltk_data...
```

```
[nltk_data]    | Downloading package udhr2 to /root/nltk_data...
[nltk_data]    |   Package udhr2 is already up-to-date!
[nltk_data]    | Downloading package unicode_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package unicode_samples 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 wordnet to /root/nltk_data...
[nltk_data]    |   Package wordnet is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_treebank_pos_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_treebank_pos_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package universal_tagset to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package universal_tagset is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package book_grammars is already up-to-date!
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package city_database is already up-to-date!
[nltk_data]    | Downloading package tagsets to /root/nltk_data...
[nltk_data]    |   Package tagsets is already up-to-date!
[nltk_data]    | Downloading package panlex_swadesh to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package panlex_swadesh is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!

[nltk_data]    |
[nltk_data]  Done downloading collection book
True
```

```
True
```

```
from nltk.corpus import wordnet
synsets = wordnet.synsets('book', pos=wordnet.NOUN)
print(synsets)
```

```
Synset('book.n.01'), Synset('book.n.02'), Synset('record.n.05'), Synset('script.n.01'), Synset('ledger.n.01'),
```

```
print("Synset: " + synsets[2].name())
print("Definition: " + synsets[2].definition())
print("Usage examples:")
for example in synsets[2].examples():
  print(example)
print("Lemmas:")
for lemma in synsets[2].lemmas():
  print(lemma.name())
print("Hierarchy traversal:")
hyp = synsets[2].hypernyms()[0]
top = wordnet.synset('entity.n.01')
while hyp:
  print(hyp)
  if hyp == top:
    break
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
Synset: record.n.05
Definition: a compilation of the known facts regarding something or someone
Usage examples:
Al Smith used to say, `Let's look at the record'
his name is in all the record books
Lemmas:
record
record_book
book
Hierarchy traversal:
Synset('fact.n.02')
Synset('information.n.01')
Synset('message.n.02')
```

```
    Synset('communication.n.02')
    Synset('abstraction.n.06')
    Synset('entity.n.01')
```

For nouns, WordNet is organizaed in such a manner that words get more and more abstract as you go up the hierarchy. The noun "record" was abstracted into "communication" and eventually "entity" of which all nouns fall under.

```
print("Hypernyms of " + synsets[2].name() + ":")
word = wordnet.synset(synsets[2].name())
hyper = lambda s: s.hypernyms()
print(list(word.closure(hyper)))

print("Hyponyms of " + synsets[2].name() + ":")
word = wordnet.synset(synsets[2].name())
hypo = lambda s: s.hyponyms()
print(list(word.closure(hypo)))

print("Meronyms of " + synsets[2].name() + ":")
word = wordnet.synset(synsets[2].name())
mero = lambda s: s.part_meronyms()
print(list(word.closure(mero)))
mero = lambda s: s.substance_meronyms()
print(list(word.closure(mero)))

print("Holonyms of " + synsets[2].name() + ":")
word = wordnet.synset(synsets[2].name())
holo = lambda s: s.part_holonyms()
print(list(word.closure(holo)))
holo = lambda s: s.substance_meronyms()
print(list(word.closure(holo)))

print("Antonyms of " + synsets[2].name() + ":")
word = wordnet.synset(synsets[2].name())
syn = list()
ant = list()
for s in wordnet.synsets(synsets[2].name()):
   for l in s.lemmas():
      ant.append(l.antonyms()[0].name())
print(str(ant))
```

```
Hypernyms of record.n.05:
[Synset('fact.n.02'), Synset('information.n.01'), Synset('message.n.02'), Synset('communication.n.02'), Synse
Hyponyms of record.n.05:
[Synset('card.n.08'), Synset('logbook.n.01'), Synset('won-lost_record.n.01'), Synset('bell_book.n.01')]
Meronyms of record.n.05:
[]
[]
Holonyms of record.n.05:
[]
[]
Antonyms of record.n.05:
[]
```

```
synsets_v = wordnet.synsets('read', pos=wordnet.VERB)
print(synsets_v)
```

```
nset('read.v.01'), Synset('read.v.02'), Synset('read.v.03'), Synset('read.v.04'), Synset('read.v.05'), Synset
```

```
print("Synset: " + synsets_v[1].name())
print("Definition: " + synsets_v[1].definition())
print("Usage examples:")
for example in synsets_v[1].examples():
  print(example)
print("Lemmas:")
for lemma in synsets_v[1].lemmas():
  print(lemma.name())
print("Hierarchy traversal:")
word = wordnet.synset(synsets_v[1].name())
hyper = lambda s: s.hypernyms()
print(list(word.closure(hyper)))
```

```
Synset: read.v.02
Definition: have or contain a certain wording or form
Usage examples:
The passage reads as follows
What does the law say?
```

```
    Lemmas:
    read
    say
    Hierarchy traversal:
    [Synset('have.v.02')]
```

Verbs have no common top-level synset, so their hierarchy traversals are much shorter. Similar to the case of nouns, as the hierarchy goes up, the verbs get more abstract.

```
print(wordnet.morphy('read', wordnet.NOUN))
print(wordnet.morphy('read', wordnet.VERB))
print(wordnet.morphy('read', wordnet.ADJ))
```

```
    read
    read
    None
```

```
apple = wordnet.synset('apple.n.01')
orange = wordnet.synset('orange.n.01')
print(wordnet.wup_similarity(apple, orange))

from nltk.wsd import lesk
sentence = ['This', 'shirt', 'is', 'orange', '.']
print(lesk(sentence, 'orange'))
```

```
    0.782608695652174
    Synset('orange.s.01')
```

WordNet found that the nouns "apple" and "orange" are ~78% similar, which makes sense as both are types of fruits. The Lesk algorithm was able to correctly identify which form of "orange" the sentence provided was using, which was the adjective form.

SentiWordNet is a library built on top of WordNet that allows the user to find the emotional meaning of a word in context. It assigns three scores: positive connotation, negative connotation, and objectivity. A possible use case for SentiWordNet would be for analyzing the user feedback of a new feature of a website.

```
from nltk.corpus import sentiwordnet
s = list(sentiwordnet.senti_synsets("best"))
print("Synsets for \"best\":")
print(s)
print("Polarity scores:")
for synset in s:
  print(synset.synset.name(), ": ", synset.pos_score(), ", ", synset.neg_score(), ", ", synset.obj_score())

sen = "this is the best day ever"
print("Sentence: " + sen)
tokens = sen.split()
for token in tokens:
  s = list(sentiwordnet.senti_synsets(token))
  if s:
    print(s[0].synset.name(), ": ", s[0].pos_score(), ", ", s[0].neg_score(), ", ", s[0].obj_score())
```

```
Synsets for "best":
[SentiSynset('best.n.01'), SentiSynset('best.n.02'), SentiSynset('best.n.03'), SentiSynset('outdo.v.02'), Sen
Polarity scores:
best.n.01 :  0.25 ,  0.0 ,  0.75
best.n.02 :  0.375 ,  0.0 ,  0.625
best.n.03 :  0.0 ,  0.0 ,  1.0
outdo.v.02 :  0.125 ,  0.0 ,  0.875
best.a.01 :  0.75 ,  0.0 ,  0.25
better.s.03 :  0.75 ,  0.0 ,  0.25
good.a.01 :  0.75 ,  0.0 ,  0.25
full.s.06 :  0.0 ,  0.0 ,  1.0
good.a.03 :  1.0 ,  0.0 ,  0.0
estimable.s.02 :  1.0 ,  0.0 ,  0.0
beneficial.s.01 :  0.625 ,  0.0 ,  0.375
good.s.06 :  1.0 ,  0.0 ,  0.0
good.s.07 :  0.75 ,  0.0 ,  0.25
adept.s.01 :  0.625 ,  0.0 ,  0.375
good.s.09 :  0.625 ,  0.0 ,  0.375
dear.s.02 :  0.5 ,  0.0 ,  0.5
dependable.s.04 :  0.5 ,  0.0 ,  0.5
good.s.12 :  0.375 ,  0.0 ,  0.625
good.s.13 :  0.625 ,  0.0 ,  0.375
effective.s.04 :  0.0 ,  0.0 ,  1.0
```

```
good.s.15 :  0.625 ,  0.0 ,  0.375
good.s.16 :  0.75 ,  0.0 ,  0.25
good.s.17 :  0.75 ,  0.0 ,  0.25
good.s.18 :  0.875 ,  0.0 ,  0.125
good.s.19 :  0.5 ,  0.0 ,  0.5
good.s.20 :  0.375 ,  0.125 ,  0.5
good.s.21 :  0.75 ,  0.0 ,  0.25
best.r.01 :  0.5 ,  0.0 ,  0.5
best.r.02 :  0.0 ,  0.0 ,  1.0
better.r.02 :  0.0 ,  0.0 ,  1.0
well.r.01 :  0.375 ,  0.0 ,  0.625
well.r.02 :  0.125 ,  0.0 ,  0.875
well.r.03 :  0.5 ,  0.0 ,  0.5
well.r.04 :  0.375 ,  0.0 ,  0.625
well.r.05 :  0.0 ,  0.0 ,  1.0
well.r.06 :  0.75 ,  0.0 ,  0.25
well.r.07 :  0.125 ,  0.0 ,  0.875
well.r.08 :  0.625 ,  0.0 ,  0.375
well.r.09 :  0.75 ,  0.0 ,  0.25
well.r.10 :  0.75 ,  0.0 ,  0.25
well.r.11 :  0.625 ,  0.0 ,  0.375
well.r.12 :  0.125 ,  0.25 ,  0.625
well.r.13 :  0.667 ,  0.333 ,  0.0
Sentence: this is the best day ever
be.v.01 :  0.25 ,  0.125 ,  0.625
best.n.01 :  0.25 ,  0.0 ,  0.75
day.n.01 :  0.0 ,  0.0 ,  1.0
ever.r.01 :  0.0 ,  0.0 ,  1.0
```

For the sentence "this is the best day ever," only some words were considered for polarity scores. "Is" had a positive score of 0.25, a negative score of 0.125, and an objective score of 0.625. I would have expected "is" to be completely objective, but it seems that is not the case. As expected, "best" has a positive score of 0.25, and an objective score of 0.75. "day" and "ever" have only objective scores of 1.0, which is unexpected. These scores are useful for NLP applications in which analyzing people's emotional reactions to something is needed.

Collocations are phrases containing words that often appear together. One or more of the words in the phrase cannot be substituted for another word and have the phrase still retain its original meaning.

```
from nltk.book import *
print("Text4 collocations:")
print(text4.collocations())
text = ' '.join(text4.tokens)
import math
vocab = len(set(text4))
hg = text.count('God bless')/vocab
print("p(God bless) = ", hg )
h = text.count('God')/vocab
print("p(God) = ", h)
g = text.count('bless')/vocab
print('p(bless) = ', g)
pmi = math.log2(hg / (h * g))
print('pmi = ', pmi)
```

```
    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
    p(God bless) =  0.0016957605985037406
    p(God) =  0.011172069825436408
    p(bless) =  0.0085785536159601
    pmi =  4.145157780720282
```

The phrase "God bless" appears in text4 17 times, while "God" appears 112 times, and "bless" appears 86 times. The phrase has a pmi score of 4.1, a positive number, which means that the phrase is likely to be a collocation.