Portfolio Assignment: Wordnet

Aditya Guin

09/25/2022

CS 4395.001

```
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('omw-1.4')
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('averaged_perceptron_tagger')
nltk.download('sentiwordnet')
from nltk.book import *
from nltk import word_tokenize, sent_tokenize, PorterStemmer, WordNetLemmatizer, pc
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn, sentiwordnet as swn
from nltk.wsd import lesk
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords 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 punkt to /root/nltk_data...
[nltk_data]   Package punkt 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 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 averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
```

1. Wordnet is a database of english words that are related by concept meaning. We call a group of these words called *synsets*, and each synset expresses a distinct concept. There are various relations in WordNet, including hypernym, hyponym, meronym, holonym, and troponym.

```
#2
noun_chosen = 'light'
synset_list = wn.synsets(noun_chosen)
print(synset_list)
```

```
[Synset('light.n.01'), Synset('light.n.02'), Synset('light.n.03'), Synset('luminosity.n.01'), Synset('light.n.05'), Syr
```

```
#3
synset_chosen = synset_list[0]

# Definition, examples, and lemmas for light
print(synset_chosen.definition())
```

```
print(synset_chosen.examples())
print(synset_chosen.lemmas())

print()
print()

# Traversing the hierarchy for light
top = wn.synset('entity.n.01')
while synset_chosen != top:
  print(synset_chosen)
  synset_chosen = synset_chosen.hypernyms()[0]

assert(top == synset_chosen)
print(synset_chosen)


synset_chosen = synset_list[0]
```

```
(physics) electromagnetic radiation that can produce a visual sensation
['the light was filtered through a soft glass window']
[Lemma('light.n.01.light'), Lemma('light.n.01.visible_light'), Lemma('light.n.01.visible_radiation')]


Synset('light.n.01')
Synset('actinic_radiation.n.01')
Synset('electromagnetic_radiation.n.01')
Synset('radiation.n.01')
Synset('energy.n.01')
Synset('physical_phenomenon.n.01')
Synset('natural_phenomenon.n.01')
Synset('phenomenon.n.01')
Synset('process.n.06')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Observation regarding hierarchy

I see that the nouns get more generalized as you go up the tree, which makes sense since we are accessing the hypernyms. Also I see that the the root of the noun will always be entity, since every noun begins at an entity. We can think of each hypernym as a parent of the current synset, and continue traversing as such.

```
# 4
print(f'Hypernyms for {noun_chosen}')
print(synset_chosen.hypernyms())
print()

print(f'Hyponyms for {noun_chosen}')
print(synset_chosen.hyponyms())
print()

print(f'Meronyms for {noun_chosen}')
print(synset_chosen.member_meronyms())
print()

print(f'Holonyms for {noun_chosen}')
print(synset_chosen.member_holonyms())
print()

print(f'Antonym for {noun_chosen}')
print(synset_chosen.lemmas()[0].antonyms())
print()
```

```
Hypernyms for light
[Synset('actinic_radiation.n.01')]

Hyponyms for light
[Synset('beam.n.04'), Synset('candlelight.n.01'), Synset('corona.n.04'), Synset('counterglow.n.01'), Synset('daylight.r

Meronyms for light
[]

Holonyms for light
[]
```

```
Antonym for light
[]
```

```
# 5
verb_chosen = 'talk'
synset_list = wn.synsets(verb_chosen)
print(synset_list)
```

```
[Synset('talk.n.01'), Synset('talk.n.02'), Synset('talk.n.03'), Synset('lecture.n.01'), Synset('talk.n.05'), Synset('ta
```

```
# 6
synset_chosen = synset_list[5]

# Definition, examples, and lemmas for talk
print(f'Definition: {synset_chosen.definition()}')
print(f'Examples: {synset_chosen.examples()}')
print(f'Lemmas: {synset_chosen.lemmas()}')

print()
print()


# Traversing the hierarchy for talk
while True:
  print(synset_chosen)
  synset_chosen = synset_chosen.hypernyms()[0]
  if len(synset_chosen.hypernyms()) == 0:
    break

print(synset_chosen)
synset_chosen = synset_list[5]
```

```
Definition: exchange thoughts; talk with
Examples: ['We often talk business', 'Actions talk louder than words']
Lemmas: [Lemma('talk.v.01.talk'), Lemma('talk.v.01.speak')]
```

```
Synset('talk.v.01')
Synset('communicate.v.02')
Synset('interact.v.01')
Synset('act.v.01')
```

Observation regarding hierarchy for *talk*

I saw that the verb talk became more generalized as we went up the tree, ultimately referring to 'act' as the most general form, which I can understand. This is different from the noun tree in that all nouns end with entity at the top of the tree.

```
verb_forms = [word for word in wn.words() if wn.morphy(word) == verb_chosen]
print(verb_forms)
```

```
['talk']
```

Observation regarding wup similarity (football and soccer)

My observation was that football and soccer had a 0.96 similarity score using wup_similarity. I expected this, because both of them are sports, and in some parts of the world, they refer to the same sport. Regarding the lesk algorithm, I expected the soccer to return as a noun, since the synset for soccer was of length 1 only.

```
#8
first_word_similar = wn.synset('football.n.01')
second_word_similar = wn.synset('soccer.n.01')

print(wn.synsets('football'))
print(wn.synsets('soccer'))


# Wu-Palmer Similarity
print(f'Wu-Palmer similarity between football and soccer: {wn.wup_similarity(first_word_similar, second_word_similar)}')
```

```
sent = ['I', 'went', 'to', 'play', 'soccer', 'with', 'my', 'friends', '.']
print(lesk(sent, 'soccer'))
```

```
[Synset('football.n.01'), Synset('football.n.02')]
[Synset('soccer.n.01')]
Wu-Palmer similarity between football and soccer: 0.96
Synset('soccer.n.01')
```

SentiWordNet is a lexical libary that assigns scores of positivity, negativity and objectivity for words. The use case for this library would be sentiment analysis of sentences. In addition, irony and sarcasm detection amongst text could use sentiwordnet

```
#9
#@title SentiWordNet is a lexical libary that assigns scores of positivity, negativity and objectivity for words. The use ca

# Polarity for all forms of antipathy
def polarity(word):
  synsets = swn.senti_synsets(word)
  for word in synsets:
    print(word)
    print(" positive score = ", word.pos_score())
    print(" negative score = ", word.neg_score())
    print(" objective score = ", word.obj_score())
    print()

emotionally_charged_word = 'lionize'
polarity(emotionally_charged_word)

print('-------------------------------------------------------------------------------')

tokens = [t.lower() for t in word_tokenize("The day is gloomy and dark, like my mind")]
print()
for t in tokens:
  polarity(t)
```

```
<judgment.n.01: PosScore=0.0 NegScore=0.0>
 positive score =  0.0
 negative score =  0.0
 objective score =  1.0

<thinker.n.01: PosScore=0.5 NegScore=0.0>
 positive score =  0.5
 negative score =  0.0
 objective score =  0.5

<mind.n.05: PosScore=0.0 NegScore=0.125>
 positive score =  0.0

 negative score =  0.125
 objective score =  0.875

<mind.n.06: PosScore=0.0 NegScore=0.0>
 positive score =  0.0
 negative score =  0.0
 objective score =  1.0

<mind.n.07: PosScore=0.375 NegScore=0.0>
 positive score =  0.375
 negative score =  0.0
 objective score =  0.625

<mind.v.01: PosScore=0.0 NegScore=0.5>
 positive score =  0.0
 negative score =  0.5
 objective score =  0.5

<mind.v.02: PosScore=0.125 NegScore=0.0>
 positive score =  0.125
 negative score =  0.0
 objective score =  0.875

<take_care.v.02: PosScore=0.0 NegScore=0.0>
 positive score =  0.0
 negative score =  0.0
 objective score =  1.0

<heed.v.01: PosScore=0.0 NegScore=0.0>
 positive score =  0.0
```

```
 positive score =  0.0
 negative score =  0.0
 objective score =  1.0

<beware.v.01: PosScore=0.5 NegScore=0.0>
 positive score =  0.5
 negative score =  0.0
 objective score =  0.5

<mind.v.06: PosScore=0.375 NegScore=0.0>
 positive score =  0.375
 negative score =  0.0


 objective score =  0.625
```

Observation regarding sentiwordnet

My observation was that the word I chose, antipathy, had a negative meaning with it. This was expected, since antipathy means deep disliking of.Words like gloomy and dark had a negative score, something I wasn't too surprised about. The applications of sentiwordnet would be sentiment analysis. This may include sarcasm and irony detection for text. If a sentiment analyzing tool is required, sentiwordnet can also be used for this.

Collocation refers to a set of words that convey more meaning that the individual words themselves. For example, old man referes to a man who is old, but old and man independently might not convey that much information. For example, old could refer to other things, and man could mean any man.

```
#10
# @title Collocation refers to a set of words that convey more meaning that the individual words themselves. For example, ol

# Collocations for text4, the Inaugural Corpus

import math
print(text4.collocations())
```

```
collocation_chosen = 'American people'

full_text4 = ' '.join(text4.tokens)
n = len(set(text4))

american_people = full_text4.count('American people') / n
american = full_text4.count('American') / n
people = full_text4.count('people') / n


print()
print("Collocation chosen: ", collocation_chosen)
# print(american_people, american, people)
print("P(American) = ", american)
print("P(people) = ", people)
print("P(American people) = ", american_people)
print("PMI = ", math.log2(american_people / (american * people)))
```

```
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

Collocation chosen:  American people
P(American) =  0.025735660847880298
P(people) =  0.06264339152119701
P(American people) =  0.00399002493765586
PMI =  1.3073947068021263
```

Observation regarding collocations

PMI is greater than 1, which means these two words (american and people) are likely to appear next to each other. This implies that these two words are a collocation, which makes sense since the inaugural corpus would refer to people as american people

✓ 0s completed at 9:04 PM