Project3-WordNet

February 24, 2023

[1]:
```
import nltk
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn
from nltk.book import *
import math

#If download needed
#nltk.download('omw-1.4')
```

```
*** Introductory Examples for the NLTK Book ***
Loading text1, …, text9 and sent1, …, sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

[2]:
```
#1 - Write a 2-3 sentence summary of WordNet
'''
WordNet is a large database of a hierarchical organization of words and their␣
 ↪relations to other words.
It has useful associated data such as Synsets which are synonyms of the word␣
 ↪(car - automobile), glosses
which are short definitions, and their relations to other words.
'''
```

[2]:
```
'\nWordNet is a large database of a hierarchical organization of words and their
relations to other words.\nIt has useful associated data such as Synsets which
are synonyms of the word (car - automobile), glosses\nwhich are short
definitions, and their relations to other words.\n'
```

[3]:
```
#2 - Select a noun. Output all synsets
noun_synset = wn.synsets('DNA')
print(noun_synset)
```

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

[4]:
```
#3 - Select one synset from the list of synsets. Extract its definition, usage␣
 ↪examples, and lemmas.
synset = noun_synset[0]

print("Definition: ", synset.definition(), '\n')
print("Examples: ", synset.examples(), '\n')
print("Lemmas: ", synset.lemmas(), "\n")

#From your selected synset, traverse up the WordNet hierarchy as far as you can␣
 ↪outputting the synsets as you go.
top = list(wn.all_synsets('n'))[0] # Find the top of the hierarchy
curr_hypernym = synset.hypernyms()[0]

print("Synsets going to the top:")
while curr_hypernym:
    print(curr_hypernym)
    if curr_hypernym == top:
        break
    if curr_hypernym.hypernyms():
        curr_hypernym = curr_hypernym.hypernyms()[0]

#Write a few sentences observing the way that WordNet is organized for nouns.
'''
For nouns the definition seems like a pretty standard explanation for what the␣
 ↪object is,
the example for the noun I used was a list with a single element string which␣
 ↪suggests there may be multiple examples for
some words. The Lemmas are all the root of dog but in different forms, with one␣
 ↪being the binomial name "Canis_familiaris".
The synsets seem to be words that are synonyms that seem to vary in relevance␣
 ↪from word to word.
'''
```

```
Definition:  (biochemistry) a long linear polymer found in the nucleus of a cell
and formed from nucleotides and shaped like a double helix; associated with the
transmission of genetic information

Examples:  ['DNA is the king of molecules']

Lemmas:  [Lemma('deoxyribonucleic_acid.n.01.deoxyribonucleic_acid'),
Lemma('deoxyribonucleic_acid.n.01.desoxyribonucleic_acid'),
Lemma('deoxyribonucleic_acid.n.01.DNA')]
```

```
Synsets going to the top:
Synset('polymer.n.01')
Synset('compound.n.02')
Synset('chemical.n.01')
Synset('material.n.01')
Synset('substance.n.01')
Synset('matter.n.03')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

[4]:
```
'\nFor nouns the definition seems like a pretty standard explanation for what
the object is,\nthe example for the noun I used was a list with a single element
string which suggests there may be multiple examples for\nsome words. The Lemmas
are all the root of dog but in different forms, with one being the binomial name
"Canis_familiaris".\nThe synsets seem to be words that are synonyms that seem to
vary in relevance from word to word.\n'
```

[5]:
```
#4 - Output the following or empty list if DNE: Hypernyms, Hyponyms, Meronyms,␣
  ↪Holonyms, Antonym
print('Hypernyms: ', synset.hypernyms(), '\n')
print('Hyponyms: ', synset.hyponyms(), '\n')
print('Meronyms: ', synset.part_meronyms(), '\n')
print('Holonyms: ', synset.part_holonyms())
```

```
Hypernyms:  [Synset('polymer.n.01')]

Hyponyms:  [Synset('complementary_dna.n.01'), Synset('episome.n.01'),
Synset('exon.n.01'), Synset('intron.n.01'), Synset('junk_dna.n.01'),
Synset('operon.n.01'), Synset('recombinant_deoxyribonucleic_acid.n.01'),
Synset('sticky_end.n.01'), Synset('transposon.n.01')]

Meronyms:  [Synset('base_pair.n.01'), Synset('gene.n.01'),
Synset('nucleic_acid.n.01')]

Holonyms:  []
```

[6]:
```
#5 - Select a verb. Output all synsets
verb_synset = wn.synsets('Thinking')
print(verb_synset)
```

```
[Synset('thinking.n.01'), Synset('think.v.01'), Synset('think.v.02'),
Synset('think.v.03'), Synset('remember.v.01'), Synset('think.v.05'),
Synset('think.v.06'), Synset('intend.v.01'), Synset('think.v.08'),
Synset('think.v.09'), Synset('think.v.10'), Synset('think.v.11'),
Synset('think.v.12'), Synset('think.v.13'), Synset('intelligent.s.04')]
```

[7]:
```
#6 - Select one syset from the list of synsets, extract its definition, usage␣
 ↪examples, and lemmas
synset = verb_synset[0]

print("Definition: ", synset.definition(), '\n')
print("Examples: ", synset.examples(), '\n')
print("Lemmas: ", synset.lemmas(), "\n")

#From your selected synset, traverse up the WordNet hierarchy as far as you can␣
 ↪outputting the synsets as you go.
curr_hypernym = synset.hypernyms()[0]

print("Synsets going to the top:")
while curr_hypernym:
    print(curr_hypernym)
    if curr_hypernym == top:
        break
    if curr_hypernym.hypernyms():
        curr_hypernym = curr_hypernym.hypernyms()[0]

#Write a few sentences observing the way that WordNet is organized for verbs.
'''
WordNet seems to be organized pretty similarly to nouns.
The definitions, examples, and lemmas fit what I would expect.
There are two examples for this verb where the noun had
one.
That may be because there's more nuance in how they're
used rather than a definable object like a noun.
'''
```

```
Definition:  the process of using your mind to consider something carefully

Examples:  ['thinking always made him frown', 'she paused for thought']

Lemmas:  [Lemma('thinking.n.01.thinking'), Lemma('thinking.n.01.thought'),
Lemma('thinking.n.01.thought_process'), Lemma('thinking.n.01.cerebration'),
Lemma('thinking.n.01.intellection'), Lemma('thinking.n.01.mentation')]

Synsets going to the top:
Synset('higher_cognitive_process.n.01')
Synset('process.n.02')
Synset('cognition.n.01')
Synset('psychological_feature.n.01')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

[7]:
```
"\nWordNet seems to be organized pretty similarly to nouns.\nThe definitions,
examples, and lemmas fit what I would expect.\nThere are two examples for this
verb where the noun had \none.\nThat may be because there's more nuance in how
they're \nused rather than a definable object like a noun.\n"
```

[8]:
```
#7 - Use morphy to find as many different forms of the word as you can
word = 'thinking'

print(wn.morphy(word))
print(wn.morphy(word, wn.ADJ))
print(wn.morphy(word, wn.VERB))
print(wn.morphy(word), wn.ADV)
print(wn.morphy(word), wn.NOUN)
```

```
thinking
thinking
think
thinking r
thinking n
```

[9]:
```
#8 - Select two words that you think might be similar. Find the specific␣
 ↪synsets you are interested in
word1 = wn.synsets('Tiger')[0]
word2 = wn.synsets('Lion')[0]

#Run the Wu-Palmer similarity metric and the Lesk algorithm
print("Wu-Palmer Similarity: ", wn.wup_similarity(word1, word2), '\n')

sent = word1.definition()
print("Lesk Algorithm: ", lesk(sent, "Lion"))

#Write a few sentences with your observations
'''
The Wu-Palmer similarity showed a pretty decent similarity between Tiger and␣
 ↪Lion at .52.
This makes sense, and the score reflects the underlying similarities between␣
 ↪the two.
For the Lesk algorithm I used the definition of a lion for my sentence and
it returned lion which follows the expected result given the inputs and their␣
 ↪overlap.
'''
```

```
Wu-Palmer Similarity:  0.5217391304347826

Lesk Algorithm:  Synset('lion.n.02')
```

[9]:
```
'\nThe Wu-Palmer similarity showed a pretty decent similarity between Tiger and
Lion at .52.\nThis makes sense, and the score reflects the underlying
```

```
similarities between the two.\nFor the Lesk algorithm I used the definition of a
lion for my sentence and\nit returned lion which follows the expected result
given the inputs and their overlap.\n'
```

[10]:
```
#9 - Write a couple of sentences about SentiWordNet, describing its␣
 ↪functionality and possible use cases
'''
SentiWordNet is built on top of WordNet and acts as a sentiment analyzer.
Given text it can try to dicern whether the tone is positive, negative or␣
 ↪neutral.
There are many possible use-cases for sentiment analysis such as chat-bots or␣
 ↪analyzing customer reviews.
'''

# Select an emotionally charged word. Find its senti-synsets and output the␣
 ↪polarity scores for each word
word = 'love'
print("Polarity scores for all senti-synsets of", word)
senti_list = list(swn.senti_synsets(word))
for item in senti_list:
    print(item)
print()

# Make up a sentence. Output the polarity for each word in the sentence
sentence = "the fox is blue and red"

print("Polarity for each word in sentence: ", sentence)

sentence = sentence.split()

for word in sentence:
    if(len(list(swn.senti_synsets(word)))) > 0:
        first_element = list(swn.senti_synsets(word))[0]
        print(first_element)


# Write a couple of sentences about your observations of the scores and the␣
 ↪utility of knowing these scores in an NLP application.
'''
It seems like the positivity and negatively vary pretty heavily based on␣
 ↪definition.
Some colors such as red it considers "negative" while some neutral like blue.
This is useful for analyzing the tone and sentiment of text, if a given text
has many negative scoring words it's more likely the tone is also negative.
'''
```

```
Polarity scores for all senti-synsets of love
```

```
<love.n.01: PosScore=0.625 NegScore=0.0>
<love.n.02: PosScore=0.375 NegScore=0.0>
<beloved.n.01: PosScore=0.125 NegScore=0.0>
<love.n.04: PosScore=0.25 NegScore=0.0>
<love.n.05: PosScore=0.0 NegScore=0.0>
<sexual_love.n.02: PosScore=0.0 NegScore=0.0>
<love.v.01: PosScore=0.5 NegScore=0.0>
<love.v.02: PosScore=1.0 NegScore=0.0>
<love.v.03: PosScore=0.625 NegScore=0.0>
<sleep_together.v.01: PosScore=0.375 NegScore=0.125>

Polarity for each word in sentence:  the fox is blue and red
<fox.n.01: PosScore=0.0 NegScore=0.0>
<be.v.01: PosScore=0.25 NegScore=0.125>
<blue.n.01: PosScore=0.0 NegScore=0.0>
<red.n.01: PosScore=0.0 NegScore=0.375>
```

[10]:
```
'\nIt seems like the positivity and negatively vary pretty heavily based on
definition.\nSome colors such as red it considers "negative" while some neutral
like blue.\nThis is useful for analyzing the tone and sentiment of text, if a
given text\nhas many negative scoring words it\'s more likely the tone is also
negative.\n'
```

[12]:
```
#10 - Write a couple of sentences about what a collocation is
'''
Collocation is how words can mean very difference things when brought together.
Even if two words are negative on their own, when combined it could result in a␣
 ↪positive statement.
It's also possible two words are a completely different noun or object when put␣
 ↪together.
'''


#  Output collocations for text4,the Inaugural corpus.
print('Collocations for text4: ')
print(text4.collocations(), '\n')

#  Select one of the collocations identified by NLTK. Calculate mutual␣
 ↪information.
text = ' '.join(text4.tokens)

words = len(set(text4))
pXY = text.count('fellow citizens')/words
print("p(fellow citizens) = ",pXY )
pX = text.count('fellow')/words
print("p(fellow) = ", pX)
pY = text.count('citizens')/words
print('p(citizens) = ', pY)
```

```
pmi = math.log2(pXY / (pX * pY))
print('pmi = ', pmi)

# Write commentary on the results of the mutual information formula and your␣
 ↪interpretation.
'''
The probability of each word seems low but is semi-frequent given the number of␣
 ↪words.
With a PMI of 4.04 that means that the probability of "fellow" being followed␣
 ↪by "citizens"
is relatively high.
This whenever we see the word "fellow" one common possible follow up word given␣
 ↪text4 would be "citizens"
'''
```

```
Collocations for text4:
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(fellow citizens) =  0.006084788029925187
p(fellow) =  0.013665835411471322
p(citizens) =  0.026932668329177057
pmi =  4.0472042737811735
```

[12]:
```
'\nThe probability of each word seems low but is semi-frequent given the number
of words.\nWith a PMI of 4.04 that means that the probability of "fellow" being
followed by "citizens"\nis relatively high. \nThis whenever we see the word
"fellow" one common possible follow up word given text4 would be "citizens"\n'
```

[ ]: