portfolio_assignment_wordnet_npj190000

February 24, 2023

Nick Johnson CS 4395.001 Portfolio Assignment: WordNet

1. Write a 2-3 sentence summary of WordNet.

WordNet is a database of English nouns, verbs, adjectives, and adverbs. These words are grouped into synonym sets, also known as synsets.

Necessary imports:

```
[25]: from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.tokenize import word_tokenize
from nltk.corpus import sentiwordnet as swn
import nltk
import math
from nltk.book import *
text4
```

```
[25]: <Text: Inaugural Address Corpus>
```

2. Output all synsets of a chosen noun. Here, I will choose the noun “book”.

```
[9]: book_synsets = wn.synsets('book')
print("All synsets of the word book: ", book_synsets)
```

```
All synsets of the word book:  [Synset('book.n.01'), Synset('book.n.02'),
Synset('record.n.05'), Synset('script.n.01'), Synset('ledger.n.01'),
Synset('book.n.06'), Synset('book.n.07'), Synset('koran.n.01'),
Synset('bible.n.01'), Synset('book.n.10'), Synset('book.n.11'),
Synset('book.v.01'), Synset('reserve.v.04'), Synset('book.v.03'),
Synset('book.v.04')]
```

3. Select one synset from the list of synsets. Extract its definition, usage examples, and lemmas. From the selected synset, traverse up the WordNet hierarchy as far as possible. Write a couple of sentences observing the way that WordNet is organized for nouns.

```
[11]: chosen_synset = book_synsets[2]
print("Chosen Synset: ", chosen_synset)
print("Definition: ", chosen_synset.definition())
print("Usage Examples: ", chosen_synset.examples())
print("Lemmas: ", chosen_synset.lemmas())
```

```
top = wn.synset('entity.n.01')
while chosen_synset:
    print(chosen_synset)
    if chosen_synset == top:
        break
    if chosen_synset.hypernyms():
        chosen_synset = chosen_synset.hypernyms()[0]
```

```
Chosen Synset:  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:  [Lemma('record.n.05.record'), Lemma('record.n.05.record_book'),
Lemma('record.n.05.book')]
Synset('record.n.05')
Synset('fact.n.02')
Synset('information.n.01')
Synset('message.n.02')
Synset('communication.n.02')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

The bottom of the hierarchy has the most specific synsets. By the time you get to the top of the hierarchy, you see the most general synsets. In this example, the most specific one is record and the most general is entity.

4. Output the hypernyms, hyponyms, meronyms, holonyms, and antonyms of the selected synset.

[12]:
```
print("Hypernyms: ", chosen_synset.hypernyms())
print("Hyponyms: ", chosen_synset.hyponyms())
print("Meronyms: ", chosen_synset.part_meronyms())
print("Holonyms: ", chosen_synset.part_holonyms())
print("Antonyms: ", chosen_synset.lemmas()[0].antonyms())
```

```
Hypernyms:  []
Hyponyms:  [Synset('abstraction.n.06'), Synset('physical_entity.n.01'),
Synset('thing.n.08')]
Meronyms:  []
Holonyms:  []
Antonyms:  []
```

5. Output all synsets of a chosen verb. Here, I will choose the verb "laugh".

[13]:
```
laugh_synsets = wn.synsets('laugh')
print("All synsets of the word laugh: ", laugh_synsets)
```

```
All synsets of the word laugh:  [Synset('laugh.n.01'), Synset('laugh.n.02'),
Synset('joke.n.01'), Synset('laugh.v.01')]
```

6. Select one synset from the list of synsets. Extract its definition, usage examples, and lemmas. From the selected synset, traverse up the WordNet hierarchy as far as possible. Write a couple of sentences observing the way that WordNet is organized for verbs.

[14]:
```python
chosen_synset = laugh_synsets[2]
print("Chosen Synset: ", chosen_synset)
print("Definition: ", chosen_synset.definition())
print("Usage Examples: ", chosen_synset.examples())
print("Lemmas: ", chosen_synset.lemmas())

top = wn.synset('entity.n.01')
while chosen_synset:
    print(chosen_synset)
    if chosen_synset == top:
        break
    if chosen_synset.hypernyms():
        chosen_synset = chosen_synset.hypernyms()[0]
```

```
Chosen Synset:  Synset('joke.n.01')
Definition:  a humorous anecdote or remark intended to provoke laughter
Usage Examples:  ['he told a very funny joke', 'he knows a million gags',
'thanks for the laugh', 'he laughed unpleasantly at his own jest', "even a
schoolboy's jape is supposed to have some ascertainable point"]
Lemmas:  [Lemma('joke.n.01.joke'), Lemma('joke.n.01.gag'),
Lemma('joke.n.01.laugh'), Lemma('joke.n.01.jest'), Lemma('joke.n.01.jape')]
Synset('joke.n.01')
Synset('wit.n.01')
Synset('message.n.02')
Synset('communication.n.02')
Synset('abstraction.n.06')
Synset('entity.n.01')
```

Similar to nouns, the bottom of the hierarchy has the most specific synsets. The top of the hierarchy has the most general synsets.

7. Use morphy to find as many different forms of the word as possible.

[20]:
```python
print(wn.morphy('laugh'))
print(wn.morphy('laughs'))
print(wn.morphy('laughing'))
print(wn.morphy('laughed'))
```

```
laugh
laugh
laugh
laugh
```

8. Select two words that you think might be similar. Find the specific synsets you are interested in. Run the Wu-Palmer similarity metric and the Lesk algorithm. Write a couple of sentences with your observations.

```
[21]: fast_synsets = wn.synsets('fast')
fast_specific = fast_synsets[3]
quick_synsets = wn.synsets('quick')
quick_specific = quick_synsets[1]

print(fast_specific)
print(fast_specific.definition())
print()
print(quick_specific)
print(quick_specific.definition())
print()

# Running Wu-Palmer Similarity Metric
print("Wu-Palmer Similarity Metric for the specific synsets: ", wn.
 ↪wup_similarity(fast_specific, quick_specific))
print()
# Running Lesk algorithm
sent = word_tokenize("The runner was moving at a quick pace.")
lesk_result = lesk(sent, 'fast')
print(lesk_result)
print(lesk_result.definition())
```

```
Synset('fast.a.01')
acting or moving or capable of acting or moving quickly

Synset('quick.s.01')
accomplished rapidly and without delay

Wu-Palmer Similarity Metric for the specific synsets:  0.5

Synset('fast.a.03')
at a rapid tempo
```

The Wu-Palmer score for the two synsets ended up being 0.5. This means they are 50% similar.

9. Write a couple of sentences about SentiWordNet, describing its functionality and possible use cases. Select an emotionally charged word. Find its senti-synsets and output the polarity scores for each word. Make up a sentence. Output the polarity for each word in the sentence. Write a couple of sentences about your observations of the scores and the utility of knowing these scores in an NLP application.

SentiWordNet is a lexical tool used to study emotion, sentiment, and opinion within text. It takes each synset and gives it three scores based on how positive, negative, and objective the term is. For use cases, SentiWordNet could be used to gauge how people are feeling when they write about products online. It could also be used with phone logs to see how clients feel when talking to customer service.

```
[22]: charged = 'happy'
list_of_sentis = list(swn.senti_synsets(charged))
```

```
print("Senti-synsets:")
for synset in list_of_sentis:
    print(synset)
    print("Positive score = ", synset.pos_score())
    print("Negative score = ", synset.neg_score())
    print("Objective score = ", synset.obj_score())
    print()
```

```
Senti-synsets:
<happy.a.01: PosScore=0.875 NegScore=0.0>
Positive score =  0.875
Negative score =  0.0
Objective score =  0.125

<felicitous.s.02: PosScore=0.75 NegScore=0.0>
Positive score =  0.75
Negative score =  0.0
Objective score =  0.25

<glad.s.02: PosScore=0.5 NegScore=0.0>
Positive score =  0.5
Negative score =  0.0
Objective score =  0.5

<happy.s.04: PosScore=0.125 NegScore=0.0>
Positive score =  0.125
Negative score =  0.0
Objective score =  0.875
```

[23]:
```
sent = "Sphinx of black quartz, judge my vow."
tokens = word_tokenize(sent)
for token in tokens:
    list_of_sentis = list(swn.senti_synsets(token))
    if list_of_sentis:
        print(list_of_sentis[0])
        print("Positive score = ", list_of_sentis[0].pos_score())
        print("Negative score = ", list_of_sentis[0].neg_score())
        print("Objective score = ", list_of_sentis[0].obj_score())
        print()
```

```
<sphinx.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0

<black.n.01: PosScore=0.0 NegScore=0.375>
Positive score =  0.0
```

```
Negative score =  0.375
Objective score =  0.625

<quartz_glass.n.01: PosScore=0.25 NegScore=0.0>
Positive score =  0.25
Negative score =  0.0
Objective score =  0.75

<judge.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0

<vow.n.01: PosScore=0.0 NegScore=0.0>
Positive score =  0.0
Negative score =  0.0
Objective score =  1.0
```

The senti-synsets for the emotionally charged word all had positive scores above 0, which I would expect given that the original word I chose was 'happy'. In terms of real life applications for these scores, I could see wanting to use this to analyze reviews of a product for example.

10. Write a couple of sentences about what a collocation is. Output collocations for text4, the Inaugural corpus. Select one of the collocations identified by NLTK. Calculate mutual information. Write commentary on the results of the mutual information formula and your interpretation.

A collocation is when a pair of words appear together more often than just by coincidence. If a pair of words is a collocation, then substituting in synonyms would cause the pair of words to lose its original meaning.

[24]:
```python
text = ' '.join(text4.tokens)
print(text4.collocations())
print()
corpus_length = len(set(text4))

# selecting 'fellow citizens' for mutual information calculation

fellow_citizens = text.count('fellow citizens') / corpus_length
print("p(fellow citizens): ", fellow_citizens)
fellow = text.count('fellow') / corpus_length
print("p(fellow): ", fellow)
citizens = text.count('citizens') / corpus_length
print("p(citizens): ", citizens)
pmi = math.log2(fellow_citizens / (fellow * citizens))
print("pmi: ", pmi)
```

```
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
```

A positive pmi means that the two words, in this case “fellow” and “citizens”, appear together more often than just by coincidence. Here, the pmi is 4.0472042737811735, which is positive, meaning that “fellow citizens” is most likely a collocation.