```
# Rick Gao - ryg180000
# CS 4395.001 - Mazidi
# WordNet Homework


# 1. WordNet is a lexical database of English words that are hierarchically grouped and connected through their semantic relationships such a
# to perform text analysis and other text-based applications in artificial intelligence.


import nltk
from nltk.corpus import wordnet
nltk.download('all')


# 2. Selecting a noun
horse_synsets = wordnet.synsets('horse')
print(horse_synsets)
```

```
[Synset('horse.n.01'), Synset('horse.n.02'), Synset('cavalry.n.01'), Synset('sawhorse.n.01'), Synset('knight.n.02'), Synset('horse.v.01
```

```
# 3. Select a synset from nouns to extract
cavalry = horse_synsets[2]
print(cavalry)
print(cavalry.definition())
print(cavalry.examples())
print(cavalry.lemmas())
```

```
Synset('cavalry.n.01')
troops trained to fight on horseback
['500 horse led the attack']
[Lemma('cavalry.n.01.cavalry'), Lemma('cavalry.n.01.horse_cavalry'), Lemma('cavalry.n.01.horse')]
```

```
# 3. Traverse WordNet hierarchy of selected synset
hypernyms = lambda x: x.hypernyms()
print(list(cavalry.closure(hypernyms)))

# For nouns, as we traverse up the WordNet hierarchy, the word becomes more general and common compared to the original word. We started with
# above it become more ambiguous, such as words like 'military personnel', 'social group', and 'group'. This makes sense since we expect the
# in the tree are less specific.
```

```
[Synset('military_personnel.n.01'), Synset('force.n.04'), Synset('organization.n.01'), Synset('social_group.n.01'), Synset('group.n.01')
```

```
# 4. Output hypernym, hyponym, meronym, holonym, antonym
print(cavalry.hypernyms())
print(cavalry.hyponyms())
print(cavalry.part_meronyms())
print(cavalry.part_holonyms())
print(cavalry.lemmas()[0].antonyms())
```

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

```
# 5. Selecting a verb and output synsets
click_synset = wordnet.synsets('click')
print(click_synset)
```

```
[Synset('chink.n.03'), Synset('suction_stop.n.01'), Synset('pawl.n.01'), Synset('click.n.04'), Synset('snap.v.04'), Synset('click.v.02')
```

```
# 6. Select a synset from verbs to extract
snap = click_synset[4]
print(snap)
print(snap.definition())
print(snap.examples())
print(snap.lemmas())
```

```
Synset('snap.v.04')
move or strike with a noise
```

```
['he clicked on the light', 'his arm was snapped forward']
[Lemma('snap.v.04.snap'), Lemma('snap.v.04.click')]
```

```
hypernyms = lambda x: x.hypernyms()
print(list(snap.closure(hypernyms)))
```

```
# Similar to nouns, it seems that as we traverse up the hierarchy, the word becomes less specific and a more ambiguous word of the original.
# is a more general verb that could be considered a parent of 'snap'.
```

```
[Synset('move.v.03')]
```

```
# 7. Find other forms of word using morphy
print(wordnet.morphy('snap'))
print(wordnet.morphy('snapping'))
print(wordnet.morphy('snapped'))
```

```
snap
snap
snap
```

```
# 8. Choose similar words and run Wu-Palmer similarity metric and Lesk algorithm
tower = wordnet.synset('tower.n.01')
building = wordnet.synset('building.n.01')

print("Wu-Palmer similarity metric for 'tower' and 'building': ", wordnet.wup_similarity(tower, building))

from nltk.wsd import lesk
sent = ["That", "building", 'is', 'tall']
print("Lesk algorithm for building: ", lesk(sent, 'building'))

sent2 = ["The", "Salesforce", "tower", "is", "the", "tallest", "in", "San Francisco"]
print("Lesk algorithm for tower: ", lesk(sent2, 'tower'))
```

```
Wu-Palmer similarity metric for 'tower' and 'building':  0.8571428571428571
Lesk algorithm for building:  Synset('building.n.04')
Lesk algorithm for tower:  Synset('column.n.04')
```

```
# SentiWordNet is a tool for determining how positive, negative, and objective a given WordNet synset is. Scores range between 0 and 1, and t
# Applications of SentiWordNet is most commonly found in opinion mining, allowing artifical intelligence to add information related to the op
# Some use cases include being able to analyze customer opinions on products and general sentinment regarding political candidates on online
```

```
# 9. Choose emotional-charged word and output polarity of each word

from nltk.corpus import sentiwordnet

senti_list = list(sentiwordnet.senti_synsets('insane'))
for item in senti_list:
    print(item)

insane = sentiwordnet.senti_synset('insane.a.01')
print(insane)
print(insane.pos_score())
print(insane.neg_score())
print(insane.obj_score())

print("-" * 25)

sent = 'You are insane for liking that'
tokens = sent.split()
for token in tokens:
    syn_list = list(sentiwordnet.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        print(f"{token} - Pos score: {syn.pos_score()} Neg score: {syn.neg_score()} Obj score: {syn.obj_score()}")
```

```
# Based on the scores of the words, we see that SentiWordNet labels 'insane' with a negative score of 0.5, and 'liking' with a score of 0.5.
# negative association in context of the sentence. Knowing these scores
```

```
<insane.a.01: PosScore=0.0 NegScore=0.5>
<harebrained.s.01: PosScore=0.375 NegScore=0.25>
<insane.a.01: PosScore=0.0 NegScore=0.5>
0.0
0.5
```

```
0.5
-------------------------
are - Pos score: 0.0 Neg score: 0.0 Obj score: 1.0
insane - Pos score: 0.0 Neg score: 0.5 Obj score: 0.5
liking - Pos score: 0.5 Neg score: 0.25 Obj score: 0.25
```

```
# Collocations describes when specific words occur together more often than normal and provide a meaning when found together that may have a
# For example, 'strong coffee' does not literally mean physically strong coffee.
```

```
# 10. Collocations

import nltk
import math
from nltk.book import *


print(text4.collocations())

print("-" * 25)

text = ' '.join(text4.tokens)
all_words = len(set(text4))

p_of_xy = text.count('American people') / all_words
print("p(American people): ", p_of_xy)
p_of_x = text.count('American') / all_words
print("p(American): ", p_of_x)
p_of_y = text.count('people') / all_words
print('p(people): ', p_of_y)
pmi = math.log2(p_of_xy / (p_of_x * p_of_y))
print('PMI: ', pmi)
```

```
# Since the PMI is positive, it indicates that the words 'American people' is indeed a collocation. I think that 'American people' can be int
# of America, regardless if they have origins from the Americas.
```

```
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(American people):  0.00399002493765586
p(American):  0.025735660847880298
p(people):  0.06264339152119701
PMI:  1.3073947068021263
```

✓ 0s completed at 3:29 PM