Portfolio Assignment: Wordnet

Ahmed Iqbal (axi180011)

Summary of Wordnet

Wordnet is a semantic database that has relations between words, such as between synonyms. Each word has a sysnet, which contains its definitions and examples of using it. This makes it a blend between a dictionary and a thesaurus, and it is availible in over 200 languages.

```
from nltk.corpus import wordnet as wn

noun = 'plane'

synsets = wn.synsets(noun)

print("Noun is", noun, "\n\nSynsets: ")
for set in synsets:
  print(set)

synset = synsets[0]

# printing the definition, examples, and lemmas of the word
print("\nDefinition of", synset.name(), ":", synset.definition())
print("Examples of", synset.name(), ":", synset.examples())
print("Lemmas:", synset.lemma_names())
print()


# getting a list of all of the items in the traveral of hieracrchy and printing them
print("Hierarchy traversal:")
lambda_func = lambda x: x.hypernyms()
traversal_list = list(synset.closure(lambda_func))
for item in traversal_list:
  print(item)

# printing the hypernyms, hyponyms, meronyms, holonyms, and antonyms
print("\nHypernyms:", synset.hypernyms())
print("Hyponyms:", synset.hyponyms())
print("Meronyms:", synset.part_meronyms())
print("Holonyms:", synset.part_holonyms())
print("Antonyms:", synset.lemmas()[0].antonyms())
```

```
Noun is plane
```

```
Synsets:
Synset('airplane.n.01')
Synset('plane.n.02')
Synset('plane.n.03')
Synset('plane.n.04')
Synset('plane.n.05')
Synset('plane.v.01')
Synset('plane.v.02')
Synset('plane.v.03')
Synset('flat.s.01')

Definition of airplane.n.01 : an aircraft that has a fixed wing and is powered by prc
Examples of airplane.n.01 : ['the flight was delayed due to trouble with the airplane
Lemmas: ['airplane', 'aeroplane', 'plane']

Hierarchy traversal:
Synset('heavier-than-air_craft.n.01')
Synset('aircraft.n.01')
Synset('craft.n.02')
Synset('vehicle.n.01')
Synset('conveyance.n.03')
Synset('instrumentality.n.03')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')

Hypernyms: [Synset('heavier-than-air_craft.n.01')]
Hyponyms: [Synset('airliner.n.01'), Synset('amphibian.n.02'), Synset('biplane.n.01'),
Meronyms: [Synset('accelerator.n.01'), Synset('escape_hatch.n.01'), Synset('fuselage.
Holonyms: []
Antonyms: []
```

How WordNet is organized for nouns

Nouns in wordnet are organized in a descending order in terms of specificity. That is, the lower down in the hierarchy a noun is, the more specific it is at describing an object. FOr example, vehicle is lower down in the hierarchy than entity.

```
from nltk.corpus import wordnet as wn

verb = 'fly'

synsets = wn.synsets(verb, pos=wn.VERB)

# printing all synsets
print("Verb is", verb, "\n\nSynsets: ")
for set in synsets:
  print(set)
```

```
# picking a synset
synset = synsets[0]

# printing the definition, examples, and lemmas of the word
print("\nDefinition of", synset.name(), ":", synset.definition())
print("Examples of", synset.name(), ":", synset.examples())
print("Lemmas: ", synset.lemma_names())

# getting a list of all of the items in the traveral of hieracrchy and printing them
print("\nHierarchy traversal:")
lambda_func = lambda x: x.hypernyms()
traversal_list = list(synset.closure(lambda_func))
for item in traversal_list:
  print(item)

# finding and printing the different forms of the word
forms_list = []
for set in synsets:
  for name in set.lemma_names():
    forms_list.append(wn.morphy(name))
print("\nDifferent forms of the word:", forms_list)
```

```
Verb is fly

Synsets:
Synset('fly.v.01')
Synset('fly.v.02')
Synset('fly.v.03')
Synset('fly.v.04')
Synset('fly.v.05')
Synset('fly.v.06')
Synset('fly.v.07')
Synset('fly.v.08')
Synset('fly.v.09')
Synset('fly.v.10')
Synset('flee.v.01')
Synset('fly.v.12')
Synset('fly.v.13')
Synset('vanish.v.05')

Definition of fly.v.01 : travel through the air; be airborne
Examples of fly.v.01 : ['Man cannot fly']
Lemmas:  ['fly', 'wing']

Hierarchy traversal:
Synset('travel.v.01')

Different forms of the word: ['fly', 'wing', 'fly', 'fly', 'aviate', 'pilot', 'fly',
```

How WordNet is organized for verbs

How WordNet is organized for verbs

Verbs in Wordnet also seem to be organized in a descending order in terms of specificity. However, unlike with the hierarchy for nouns, there are not many results for my chosen verb or some others that I have tried, so my observations hold less weight.

```
from nltk.wsd import lesk
from nltk.corpus import wordnet as wn

word1, word2 = "snow", "frost"
print(f"The two words are {word1} and {word2}")

# getting synset for the two words
synset1 = wn.synsets(word1)[0]
synset2 = wn.synsets(word2)[0]

# calculating the wu-palmer metric over the two synsets
print(f"\nThe Wu-Palmer similarity metric: {wn.wup_similarity(synset1, synset2)}")

# splitting example sentences
sent1, sent2 = "The snow is falling heavily", "The morning frost is everywhere"
split_sent1 = sent1.split(" ")
split_sent2 = sent2.split(" ")

# running the lesk algorithm for both words over their respective sentences
lesk1 = lesk(split_sent1, word1)
lesk2 = lesk(split_sent2, word2)

# printing the output of the lesk algorithm
print(f"\nThe output of the lesk algorithm for \"{word1}\" over \"{sent1}\" is {lesk1} and
```

```
     The two words are snow and frost

     The Wu-Palmer similarity metric: 0.23529411764705882

     The output of the lesk algorithm for "snow" over "The snow is falling heavily" is Syr
```

Observations on Wu-Palmer similarity metric and Lesk algorithm

From our results, we can see that there a low relationship between snow and frost, which is quite suprising. Since it calculates relatedness using the depths of the words in the hierarchy, I thought the score would be higher. As far as the Lesk algorithm goes, it seems to correctly indentify which synset of a word is being used in a given sentence.

which synset of a word is being used in a given sentence.

About SentiWordNet

SentiWordNet is a lexical resource that is built on top of WordNet. It can assign positive, negative, and overall scores to synsets and and analyze a text for positive/negative or objective/subjective tones. Thus, it is able to capture the human sentiments that are covneyed with the words, not just the dictionary definitions.

```
import nltk
nltk.download('sentiwordnet')
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.corpus import sentiwordnet as swn

emotional = "passionate"

# finding all synsets for "passionate"
synsets = swn.senti_synsets(emotional)

# printing scores of all synsets
for synset in synsets:
  print(f"\nThe word {synset.synset.name()} has a positive score of {synset.pos_score()}, a

sent = "You would be hard pressed to find a man more passionate about a cause."
split_sent = sent.split(" ")

# printing the polarity for each word in the example sentence
for word in split_sent:
  try:
    synset = list(swn.senti_synsets(word))[0]
    print(f"The polarity of {word} is {synset.pos_score() - synset.neg_score()}")
  except:
    print(f"There is no synset for {word}")

print()
```

```
    The word passionate.a.01 has a positive score of 0.375, a negative score of 0.375, ar

    There is no synset for You
    There is no synset for would
    The polarity of be is 0.0
    The polarity of hard is -0.75
    The polarity of pressed is 0.0
```

```
There is no synset for to
The polarity of find is 0.625
The polarity of a is 0.0
The polarity of man is 0.0
The polarity of more is 0.0
The polarity of passionate is 0.0
The polarity of about is 0.0
The polarity of a is 0.0
There is no synset for cause.

[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Package sentiwordnet 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 omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

Utility in NLP application

The ability to know the tone or objectiveness of a piece of text is incredibly useful. It can be used to filter news by analyzing how biased it is, or it could be used to combat cyberbullying by looking at the sentiments in addition to commonly used speech in bullying.

About collocations

A collocation is essentially a series of words that occurs in a predictable manner and often enough that it is not by chance alone. They can be made up of any kind of word. An example is saying something has a "rich history". We would not say "wealthy history" because it doesn't have the same meaning colloquially and it sounds off.

```
import math
from nltk.text import Text
from nltk.corpus import inaugural
nltk.download('inaugural')
nltk.download('stopwords')
nltk.download('punkt')

# getting text4 and a collocation
text4 = Text(inaugural.words())
collocation = list(text4.collocation_list())[0]

# printing the collocation
print(f"\nCollocation: {collocation}")

# finding the number f tokens in the text
```

```
tokens = nltk.word_tokenize(inaugural.raw())
num_of_tokens = len(tokens)

# combining the collocation and tokens into strings to search through them easier
text4 = " ".join(tokens)
colloc = " ".join(collocation)

# caluclating and printing the pmi according to the equation log2(P(x,y)/(P(x)*P(y)))
print(f"The PMI is {math.log2((text4.count(colloc)/num_of_tokens)/((text4.count(collocatio
```

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

Collocation: ('United', 'States')
The PMI is 8.738235157306907
```

Results of mutual information formula

Based on the result of 8.74 from the mutual information formula, we can safely say that there is a strong correlation between the words "United" and "States". This is because the number is positive and quite high. Had it been zero, we would say that there is no correlation. And had the number been negative, we would say that there is a anegative correlation. Thus, put together, they are very likely to be a collocation.