WordNet

CS 4395 - Intro to NLP

Dr. Karen Mazidi

Prepare by Leo Nguyen - ldn190002

Instruction 1

Summary of WordNet

WordNet is a lexical database of English. Nouns, verbs, adjectives and adverbs are organized into a hierarchy of synsets. A synset is a cognitive synonym set, each expressing a distinct concept.

Intruction 2

Import WordNet

In [1]:
```
from nltk.corpus import wordnet as wn
```

Get all synsets of noun 'book'

In [2]:
```
wn.synsets('house')
```

Out[2]:
```
[Synset('house.n.01'),
 Synset('firm.n.01'),
 Synset('house.n.03'),
 Synset('house.n.04'),
 Synset('house.n.05'),
 Synset('house.n.06'),
 Synset('house.n.07'),
 Synset('sign_of_the_zodiac.n.01'),
 Synset('house.n.09'),
 Synset('family.n.01'),
 Synset('theater.n.01'),
 Synset('house.n.12'),
 Synset('house.v.01'),
 Synset('house.v.02')]
```

Instruction 3

From selected synset - **Synset('house.n.01')** we can extract:

```
- Definition
- Usage examples
- Lemmas
```

Extract definition

```
In [3]: house = wn.synset('house.n.01') # Assign the selected synset to house
        house.definition()
```

```
Out[3]: 'a dwelling that serves as living quarters for one or more families'
```

Extract usage examples

```
In [4]: house.examples()
```

```
Out[4]: ['he has a house on Cape Cod', 'she felt she had to get out of the house']
```

Extract lemmas

```
In [5]: house.lemmas()
```

```
Out[5]: [Lemma('house.n.01.house')]
```

Traverse up the WordNet hierarchy for noun

The hierarchy for nouns has 'entity' at the top.

As the selected synset above is noun. So it's top level is 'entity'.

We can verify that with the code below

```
In [6]: house.root_hypernyms()
```

```
Out[6]: [Synset('entity.n.01')]
```

The code below traverse up the WordNet hierarchy of selected synset and print out each level as we go.

The loop stop when it reach the top level which is 'entity'

```
In [7]: hyp = house.hypernyms()[0]
        top = wn.synset('entity.n.01')
        while hyp:
            print(hyp)
            if hyp == top:
                break
            if hyp.hypernyms():
                hyp = hyp.hypernyms()[0]
```

```
Synset('building.n.01')
Synset('structure.n.01')
Synset('artifact.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

Observation on WordNet hierarchy for nouns

- The WordNet hierarchy for nouns was organized into different levels. The upper levels are more general and more abstract concepts compare to the lower levels. The top of the hierarchy for nouns is 'entity'

Intruction 4

For each synset we can extract:

```
- Hypernyms (higher)
- Hyponyms (lower)
- Meronyms (part of)
- Holonyms (whole)
- Antonyms (opposite)
```

Hypernyms: higher, more general concept.

The code below output the house's hypernyms

```
In [8]: house.hypernyms()
```

```
Out[8]: [Synset('building.n.01'), Synset('dwelling.n.01')]
```

Hyponyms : lower, more specific concept

The code below output the house's hyponyms

```
In [9]: house.hyponyms()
```

```
Out[9]: [Synset('beach_house.n.01'),
         Synset('boarding_house.n.01'),
         Synset('bungalow.n.01'),
         Synset('cabin.n.02'),
         Synset('chalet.n.01'),
         Synset('chapterhouse.n.02'),
         Synset('country_house.n.01'),
         Synset('detached_house.n.01'),
         Synset('dollhouse.n.01'),
         Synset('duplex_house.n.01'),
         Synset('farmhouse.n.01'),
         Synset('gatehouse.n.01'),
         Synset('guesthouse.n.01'),
         Synset('hacienda.n.02'),
         Synset('lodge.n.04'),
         Synset('lodging_house.n.01'),
         Synset('maisonette.n.02'),
         Synset('mansion.n.02'),
         Synset('ranch_house.n.01'),
         Synset('residence.n.02'),
         Synset('row_house.n.01'),
         Synset('safe_house.n.01'),
         Synset('saltbox.n.01'),
         Synset('sod_house.n.01'),
         Synset('solar_house.n.01'),
         Synset('tract_house.n.01'),
         Synset('villa.n.02')]
```

Meronyms: a part of something. It coud be:

```
- part_meronyms(): by parts
- substance_meronyms(): by substances
```

The code below output the house's meronyms by parts

```
In [10]: house.part_meronyms()
```

```
Out[10]: [Synset('library.n.01'),
          Synset('loft.n.02'),
          Synset('porch.n.01'),
          Synset('study.n.05')]
```

The code below output the house's meronyms by substance.

It is empty because house is not subtances of anything

```
In [11]: house.substance_meronyms()
```

```
Out[11]: []
```

Holonyms: a whole of something. It could be:

```
- part_holonyms(): by parts
- substance_holonyms(): by substances
```

The code below output the house's holonyms by parts

```
In [12]: house.part_holonyms()
```

```
Out[12]: []
```

The code below output the house's holonyms by substances

```
In [13]: house.part_holonyms()
```

```
Out[13]: []
```

Antonyms: oppostite of something.

The code below output the house's antonyms.

The list is empty even we try it the lemmas() function.

This mostly happen with nouns. We can easily find the antonyms with verbs or adjectives

```
In [14]: [lemma.antonyms() for lemma in house.lemmas()]
```

```
Out[14]: [[]]
```

Instruction 5

Get all synsets of verb 'run'

In [15]:
```
wn.synsets('run')
```

```
Out[15]: [Synset('run.n.01'),
          Synset('test.n.05'),
          Synset('footrace.n.01'),
          Synset('streak.n.01'),
          Synset('run.n.05'),
          Synset('run.n.06'),
          Synset('run.n.07'),
          Synset('run.n.08'),
          Synset('run.n.09'),
          Synset('run.n.10'),
          Synset('rivulet.n.01'),
          Synset('political_campaign.n.01'),
          Synset('run.n.13'),
          Synset('discharge.n.06'),
          Synset('run.n.15'),
          Synset('run.n.16'),
          Synset('run.v.01'),
          Synset('scat.v.01'),
          Synset('run.v.03'),
          Synset('operate.v.01'),
          Synset('run.v.05'),
          Synset('run.v.06'),
          Synset('function.v.01'),
          Synset('range.v.01'),
          Synset('campaign.v.01'),
          Synset('play.v.18'),
          Synset('run.v.11'),
          Synset('tend.v.01'),
          Synset('run.v.13'),
          Synset('run.v.14'),
          Synset('run.v.15'),
          Synset('run.v.16'),
          Synset('prevail.v.03'),
          Synset('run.v.18'),
          Synset('run.v.19'),
          Synset('carry.v.15'),
          Synset('run.v.21'),
          Synset('guide.v.05'),
          Synset('run.v.23'),
          Synset('run.v.24'),
          Synset('run.v.25'),
          Synset('run.v.26'),
          Synset('run.v.27'),
          Synset('run.v.28'),
          Synset('run.v.29'),
          Synset('run.v.30'),
          Synset('run.v.31'),
          Synset('run.v.32'),
          Synset('run.v.33'),
          Synset('run.v.34'),
          Synset('ply.v.03'),
          Synset('hunt.v.01'),
          Synset('race.v.02'),
          Synset('move.v.13'),
          Synset('melt.v.01'),
          Synset('ladder.v.01'),
          Synset('run.v.41')]
```

Instruction 6

From selected synset - **Synset('run.v.01')** we can extract:

```
- Definition
- Usage examples
- Lemmas
```

Extract definition

```
In [16]: run = wn.synset('run.v.01')  # Assign selected synset to run
         run.definition()
```

```
Out[16]: "move fast by using one's feet, with one foot off the ground at any given tim
         e"
```

Extract usage examples

```
In [17]: run.examples()
```

```
Out[17]: ["Don't run--you'll be out of breath", 'The children ran to the store']
```

Extract lemmas

```
In [18]: run.lemmas()
```

```
Out[18]: [Lemma('run.v.01.run')]
```

Traverse up the WordNet hierarchy for verb

```
In [19]: hyper = lambda s: s.hypernyms()
         list(run.closure(hyper))
```

```
Out[19]: [Synset('travel_rapidly.v.01'), Synset('travel.v.01')]
```

Observation on WordNet hierarchy for verb

```
- The WordNet hierarchy for verb is different from nouns. There is no top level for
verb's hierarchy.
```

Instruction 7

Morphy: mean base form of the word.

We can use morphy() function to do so

The code below find the different form of the word **'interest'** by:

```
- Get all the English words
- Go through each word and get it base form. Then compare it the target word.
- If it is the same, add the the list forms
- Remove the duplicate by using set()
```

In [20]:
```
from nltk.corpus import words

en_words = words.words()  # Get all English words
forms = []
word = 'interest'

for w in en_words:
    if wn.morphy(w) == word: # Check if the word has the same base form of the
target word
        forms.append(w)

forms = set(forms) # Remove duplicate words
print('Different forms of the word - interest:', forms)
```

```
Different forms of the word - interest: {'interest', 'interested', 'interesti
ng'}
```

Instruction 8

Word Similarity:

- We can find the similarity score between 2 word by using the Wu-Palmer metric or the Lesk algorithm.

```
 - The Wu-Palmer imilarity score range from 0 (little similarity) to 1 (identit
y)
 - The Lesk algorithm using Word-sense disambiguation (WSD) to find the synset
 with the highest number of overlapping words between the context sentence and
 the defitions in each synset for target word.
```

We will compare **'jet'** and **'helicopter'** using 2 algorithm above.

Check similarity using **Wu-Palmer Metric**

First let check the definition of each word for specific synset that we use

Case 1: Both have the same part of speech (POS)

```
In [21]: jet_noun= wn.synset('jet.n.01') # For
         print('Jet definition ( as a noun): ', jet_noun.definition())

         helicopter_noun = wn.synset('helicopter.n.01')
         print('Helicopter definition (as a noun): ', helicopter_noun.definition())
```

```
Jet definition ( as a noun):  an airplane powered by one or more jet engines
Helicopter definition (as a noun):  an aircraft without wings that obtains it
s lift from the rotation of overhead blades
```

Output the Wu-Palmer score (Same POS, both words are noun)

```
In [22]: wn.wup_similarity(jet_noun,helicopter_noun)
```

```
Out[22]: 0.88
```

Case 2: Two word have different POS

```
In [23]: jet_verb= wn.synset('jet.v.01')
         print('Jet definition ( as a verb): ', jet_verb.definition())

         helicopter_noun = wn.synset('helicopter.n.01')
         print('Helicopter definition (as a noun): ', helicopter_noun.definition())
```

```
Jet definition ( as a verb):  issue in a jet; come out in a jet; stream or sp
ring forth
Helicopter definition (as a noun):  an aircraft without wings that obtains it
s lift from the rotation of overhead blades
```

Output the Wu-Palmer score(Different POS, Jet is verb, Helicoprer is noun

```
In [24]: wn.wup_similarity(jet_verb,helicopter_noun)
```

```
Out[24]: 0.11764705882352941
```

Check similarity using **Lesk Algorithm**

```
In [25]: from nltk.wsd import lesk # Importing the Leak before using it
```

First, we look at all the definition for word **'jet'**

```
In [26]: for ss in wn.synsets('jet'):
             print(ss, ss.definition())
```

```
Synset('jet.n.01') an airplane powered by one or more jet engines
Synset('jet.n.02') the occurrence of a sudden discharge (as of liquid)
Synset('jet.n.03') a hard black form of lignite that takes a brilliant polish
and is used in jewelry or ornamentation
Synset('jet.n.04') atmospheric discharges (lasting 10 msec) bursting from the
tops of giant storm clouds in blue cones that widen as they flash upward
Synset('k.n.07') street names for ketamine
Synset('fountain.n.03') an artificially produced flow of water
Synset('jet.v.01') issue in a jet; come out in a jet; stream or spring forth
Synset('jet.v.02') fly a jet plane
Synset('coal-black.s.01') of the blackest black; similar to the color of jet
or coal
```

As, we can see, there are many different definition for the word 'jet' depend on which synset that we choose.

We can find which synset has the best definition for contect text below:

```
"A water jet cutter, also known as a water jet or waterjet, is an industrial tool c
apable of cutting a wide variety of materials using an extremely high-pressure jet
 of water, or a mixture of water and an abrasive substance."
```

The code below tokenize the text into word and save to list tokens.

```
In [27]: from nltk import word_tokenize
         text = "A water jet cutter, also known as a water jet or waterjet, is an indus
         trial tool capable of cutting a wide variety of materials using an extremely h
         igh-pressure jet of water, or a mixture of water and an abrasive substance."
         tokens = word_tokenize(text)
```

Using Lesk algoritm without specify the part of speech (POS) to use.

As we can that, the lesk did not do a good job. The output synset does not have the definition match with text's context

```
In [28]: syn1 = lesk(tokens, 'jet')
         print("Definition of", syn1, ':', syn1.definition())
```

```
Definition of Synset('jet.n.03') : a hard black form of lignite that takes a
brilliant polish and is used in jewelry or ornamentation
```

Using Lesk algoritm with specify the part of speech (POS) to use.

As we can see, the lesk also not perform a good job even though we specify the pos

In [29]:
```
syn2 = lesk(tokens, 'jet', 'v')
print("Definition of", syn2, ':', syn2.definition())
```

```
Definition of Synset('jet.v.02') : fly a jet plane
```

In [30]:
```
syn3 = lesk(tokens, 'jet', 'n')
print("Definition of", syn3, ':', syn3.definition())
```

```
Definition of Synset('jet.n.03') : a hard black form of lignite that takes a
brilliant polish and is used in jewelry or ornamentation
```

Observation on Wu-Palmer and Lesk Algorithm

- The Wu-Palmer output is score. The result is more accurate if the selected synset of both words have the same POS compare when they are not.
- The Lesk Algorithm output is a synset. There is still a lot improvement to make on the Lesk Algorithm. As we can see in the example above, even though we specifiy the POS, the algorithm could not produce the correct answer. It is because, it compare the target word with context, and the context could be anything not like in the Wu-Palmer - a built database as WordNet hierarchy

Instruction 9

SentiWordNet is a lexical resourse for opinion minining. It divides sentiment scores of each synset into 3 classifications: positivity, negativity, and objectivity with the sum of 3 values is always 1. It was built on top of WordNet. The SentiWordNet can be used check the popularity of a product, people by analyzing the sentiment level on reviews, comments about the product, people on multimedia platform such as: Twitter, Instagram, etc.

Importing the SentiWordNet

In [31]:
```
from nltk.corpus import sentiwordnet as swn
from nltk.corpus import stopwords
```

The code below find a list of senti-synsets of the the word **'fast'**, and print out the polarity score for each synset

In [32]:
```
senti_list = list(swn.senti_synsets('fast'))
for item in senti_list:
    print(item)
```

```
<fast.n.01: PosScore=0.0 NegScore=0.0>
<fast.v.01: PosScore=0.0 NegScore=0.0>
<fast.v.02: PosScore=0.0 NegScore=0.0>
<fast.a.01: PosScore=0.0 NegScore=0.0>
<fast.a.02: PosScore=0.0 NegScore=0.0>
<fast.a.03: PosScore=0.0 NegScore=0.0>
<fast.s.04: PosScore=0.0 NegScore=0.0>
<fast.s.05: PosScore=0.125 NegScore=0.0>
<debauched.s.01: PosScore=0.375 NegScore=0.625>
<flying.s.02: PosScore=0.25 NegScore=0.25>
<fast.s.08: PosScore=0.0 NegScore=0.0>
<firm.s.10: PosScore=0.5 NegScore=0.0>
<fast.s.10: PosScore=0.25 NegScore=0.0>
<fast.r.01: PosScore=0.0 NegScore=0.0>
<fast.r.02: PosScore=0.0 NegScore=0.0>
```

The polarity of the word 'fast'

In [33]:
```
pos = 0
neg = 0
obj = 0

for item in senti_list:
    pos += item.pos_score()
    neg += item.neg_score()
    obj += item.obj_score()
print("The polarity score of word 'fast': ")
pos = "%.2f" % (pos/len(senti_list))
neg = "%.2f" % (neg/len(senti_list))
obj = "%.2f" % (obj/len(senti_list))
print('Positive:', pos)
print('Negative:', neg)
print('Objective:', obj)
```

```
The polarity score of word 'fast':
Positive: 0.10
Negative: 0.06
Objective: 0.84
```

Output the polarity for each word in the sentence.

```
- Before we can calculate the polarity, we need to remove number, stopword and punc
tuation in the text
- Make unique list, so we only output each word once
```

In [34]:
```python
senti_list = list(swn.senti_synsets('the'))
for item in senti_list:
    print(item)
```

In [35]:
```python
sent = 'Today is a beautiful day'

tokens = word_tokenize(sent)

for token in tokens:
    senti_list = list(swn.senti_synsets(token))
    pos = 0
    neg = 0
    obj = 0

    for item in senti_list:
        pos += item.pos_score()
        neg += item.neg_score()
        obj += item.obj_score()
    print("\nThe polarity score of word -",token, "-")
    pos = "%.2f" % (pos/len(senti_list))
    neg = "%.2f" % (neg/len(senti_list))
    obj = "%.2f" % (obj/len(senti_list))
    print('Positive:', pos)
    print('Negative:', neg)
    print('Objective:', obj)
```

```
The polarity score of word - Today -
Positive: 0.09
Negative: 0.00
Objective: 0.91

The polarity score of word - is -
Positive: 0.03
Negative: 0.02
Objective: 0.95

The polarity score of word - a -
Positive: 0.02
Negative: 0.04
Objective: 0.95

The polarity score of word - beautiful -
Positive: 0.69
Negative: 0.00
Objective: 0.31

The polarity score of word - day -
Positive: 0.05
Negative: 0.00
Objective: 0.95
```

Observation for SentiWordNet

```
- In general, if the word is noun or verb, it will properly get the high objective
 score. By knowing the polarity score of each word in the sentences (context), we c
an use them to calculate the general sentiment of the whole context.
```

Instruction 10

Collocations are words that usually appear together to form a greater meaning or unique concept than the sum of their parts. It is because there are many case that a group of word usually appear together but do not provide greater meaning such as 'of the'.

Import the book objects

In [36]:
```python
from nltk.book import *
```

```
*** 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
```

Output collocations for text4

In [37]:
```python
text4.collocations()
```

```
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
```

Calculating the pointwise mutual information (PMI) for **"United States"** and **'of the'**

For **United States**

```
In [38]: text = ' '.join(text4.tokens)

         import math
         vocab = len(set(text4))
         us = text.count('United States')/vocab
         print("p(United States) = ",us )
         u = text.count('United')/vocab
         print("p(United) = ", u)
         s = text.count('States')/vocab
         print("p(States) = ", s)
         pmi = math.log2(us / (u * s))
         print('pmi = ', pmi)
```

```
p(United States) =  0.015860349127182045
p(United) =  0.0170573566084788
p(States) =  0.03301745635910224
pmi =  4.815657649820885
```

For **of the**

```
In [39]: hg = text.count('of the')/vocab
         print("p(of the) = ",hg )
         o = text.count('of')/vocab
         print("p(of) = ", o)
         t = text.count('the ')/vocab # space so it doesn't capture 'their' etc.
         print('p(the) = ', t)
         pmi = math.log2(hg / (o * t))
         print('pmi = ', pmi)
```

```
p(of the) =  0.20089775561097256
p(of) =  0.7487281795511221
p(the) =  0.9533167082294264
pmi =  -1.8290080938996587
```

Comments and Interpretation

```
- If a group of words has a higher score on PMI, it is more likely to be a collocat
ion. This can be see clear when comparing the PMI between 'United States' and 'of t
he'.
- The PMI based on the idea that if a group of words has the high join probability,
and some of them, or all of them have the high probabilty seperately. It is more li
kely that the group was formed by chance, and not convey any special meaning. Howev
er, of each have high probability and the join probabilty is low. It mean, there mu
st be a reason for them to stay close together, to create a higher meaning.
```