Motalib Rahim mxr170012

1.

WordNet

WordNet is an organized hierarchical structure based on the features of the English language. It is a great source for acquiring information regarding how concepts are organized hierarchically. Such was the interest of George Miller who started the project.

```
# requires to download packages
import nltk
from nltk.corpus import wordnet as wn # NLTK - python interface for WordNet
```

```
#2.
wn.synsets('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.
wn.synset('record.n.05').definition()
```

```
'a compilation of the known facts regarding something or someone'
```

```
wn.synset('record.n.05').examples()
```

```
["Al Smith used to say, `Let's look at the record'",
 'his name is in all the record books']
```

```
wn.synset('record.n.05').lemmas()
```

✓ 0s completed at 2:56 AM

```
[Lemma('record.n.05.record'),
 Lemma('record.n.05.record_book'),
 Lemma('record.n.05.book')]
```

```
book_synsets = wn.synsets('record', pos=wn.NOUN)
for sense in book_synsets:
    lemmas = [l.name() for l in sense.lemmas()]
    print("Synset: " + sense.name() + "(" +sense.definition() + ")  \n\t Lemmas:" + str(ler
```

```
Synset: record.n.01(anything (such as a document or a phonograph record or a photogra
         Lemmas:['record']
Synset: phonograph_record.n.01(sound recording consisting of a disk with a continuous
         Lemmas:['phonograph_record', 'phonograph_recording', 'record', 'disk', 'disc
Synset: record.n.03(the number of wins versus losses and ties a team has had)
         Lemmas:['record']
Synset: record.n.04(the sum of recognized accomplishments)
         Lemmas:['record', 'track_record']
Synset: record.n.05(a compilation of the known facts regarding something or someone)
         Lemmas:['record', 'record_book', 'book']
Synset: record.n.06(an extreme attainment; the best (or worst) performance ever attes
         Lemmas:['record']
Synset: record.n.07(a document that can serve as legal evidence of a transaction)
         Lemmas:['record']
Synset: criminal_record.n.01(a list of crimes for which an accused person has been pr
         Lemmas:['criminal_record', 'record']
```

Observation of WordNet Organization

Working with WordNet gives information regarding how for a word the synsets are organized. It reads in the semantic quality of the word and categorize at first with the basic qualities and then onto more in depth. This eases the method of interpreting the context of a specific word.

```
#4.
book = wn.synset('record.n.05')
print('hypernyms: ', book.hypernyms())
print('hyponyms: ', book.hyponyms())
```

```
hypernyms:  [Synset('fact.n.02')]
hyponyms:  [Synset('card.n.08'), Synset('logbook.n.01'), Synset('won-lost_record.n.01
```

```
meronyms = book.part_meronyms()  # get the parts/meronyms of the word
print(f"Meronyms: '{book}': {meronyms}")
```

```
Meronyms: 'Synset('record.n.05')': []
```

```
holonyms = book.part_holonyms()
```

```
print(f"Holonyms: '{book}': {holonyms}")
```

```
Holonyms: 'Synset('record.n.05')': []
```

```
noun = 'book'
synsets = wn.synsets(noun)
antonyms = ()
for synset in synsets:
    for lemma in synset.lemmas():
        for antonym in lemma.antonyms():
            antonyms.add(antonym.name())
print(f"Antonyms of '{noun}': {antonyms}")
```

```
Antonyms of 'book': ()
```

```
#5.
wn.synsets('run')
```

```
[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')]
```

```
#6.
wn.synset('run.v.01').definition()
```

```
'move fast by using one's feet, with one foot off the ground at any given time'
```

```
wn.synset('run.v.01').examples()
```

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

```
wn.synset('run.v.01').lemmas()
```

```
[Lemma('run.v.01.run')]
```

```
run_synsets = wn.synsets('run', pos=wn.VERB)
for sense in run_synsets:
    lemmas = [l.name() for l in sense.lemmas()]
    print("Synset: " + sense.name() + "(" +sense.definition() + ")  \n\t Lemmas:" + str(ler
```

```
 Synset: run.v.01(move fast by using one's feet, with one foot off the ground at any g
          Lemmas:['run']
 Synset: scat.v.01(flee; take to one's heels; cut and run)
          Lemmas:['scat', 'run', 'scarper', 'turn_tail', 'lam', 'run_away', 'hightail_
 Synset: run.v.03(stretch out over a distance, space, time, or scope; run or extend be
          Lemmas:['run', 'go', 'pass', 'lead', 'extend']
 Synset: operate.v.01(direct or control; projects, businesses, etc.)
          Lemmas:['operate', 'run']
 Synset: run.v.05(have a particular form)
```

```
         Lemmas:['run', 'go']
Synset: run.v.06(move along, of liquids)
         Lemmas:['run', 'flow', 'feed', 'course']
Synset: function.v.01(perform as expected when applied)
         Lemmas:['function', 'work', 'operate', 'go', 'run']
Synset: range.v.01(change or be different within limits)
         Lemmas:['range', 'run']
Synset: campaign.v.01(run, stand, or compete for an office or a position)
         Lemmas:['campaign', 'run']
Synset: play.v.18(cause to emit recorded audio or video)
         Lemmas:['play', 'run']
Synset: run.v.11(move about freely and without restraint, or act as if running around
         Lemmas:['run']
Synset: tend.v.01(have a tendency or disposition to do or be something; be inclined)
         Lemmas:['tend', 'be_given', 'lean', 'incline', 'run']
Synset: run.v.13(be operating, running or functioning)
         Lemmas:['run']
Synset: run.v.14(change from one state to another)
         Lemmas:['run']
Synset: run.v.15(cause to perform)
         Lemmas:['run']
Synset: run.v.16(be affected by; be subjected to)
         Lemmas:['run']
Synset: prevail.v.03(continue to exist)
         Lemmas:['prevail', 'persist', 'die_hard', 'run', 'endure']
Synset: run.v.18(occur persistently)
         Lemmas:['run']
Synset: run.v.19(carry out a process or program, as on a computer or a machine)
         Lemmas:['run', 'execute']
Synset: carry.v.15(include as the content; broadcast or publicize)
         Lemmas:['carry', 'run']
Synset: run.v.21(carry out)
         Lemmas:['run']
Synset: guide.v.05(pass over, across, or through)
         Lemmas:['guide', 'run', 'draw', 'pass']
Synset: run.v.23(cause something to pass or lead somewhere)
         Lemmas:['run', 'lead']
Synset: run.v.24(make without a miss)
         Lemmas:['run']
Synset: run.v.25(deal in illegally, such as arms or liquor)
         Lemmas:['run', 'black_market']
Synset: run.v.26(cause an animal to move fast)
         Lemmas:['run']
Synset: run.v.27(be diffused)
         Lemmas:['run', 'bleed']
Synset: run.v.28(sail before the wind)
         Lemmas:['run']
Synset: run.v.29(cover by running; run a certain distance)
         Lemmas:['run']
```

How WordNet is organized for verb

Similarly like the noun the verb is also semantically read by the WordNet. It gives an insight of

how the verb is being used and its relation. It also eases the method to interpret the context of the word.

```
#7.
wn.morphy('running', wn.VERB)
```

```
'run'
```

```
wn.morphy('runniest')
```

```
'runny'
```

```
wn.morphy('running', wn.ADJ)
```

```
'running'
```

```
#8.
run = wn.synset('run.v.05')
sprint = wn.synset('sprint.v.01')
dash = wn.synset('dash.v.01')
```

```
wn.wup_similarity(run, sprint)
```

```
0.25
```

```
wn.wup_similarity(run, dash)
```

```
0.2857142857142857
```

```
from nltk.wsd import lesk
for ss in wn.synsets('run'):
  print(ss, ss.definition())
```

```
Synset('run.n.01') a score in baseball made by a runner touching all four bases safel
Synset('test.n.05') the act of testing something
Synset('footrace.n.01') a race run on foot
Synset('streak.n.01') an unbroken series of events
Synset('run.n.05') (American football) a play in which a player attempts to carry the
Synset('run.n.06') a regular trip
Synset('run.n.07') the act of running; traveling on foot at a fast pace
Synset('run.n.08') the continuous period of time during which something (a machine or
Synset('run.n.09') unrestricted freedom to use
Synset('run.n.10') the production achieved during a continuous period of operation (o
Synset('rivulet.n.01') a small stream
Synset('political_campaign.n.01') a race between candidates for elective office
Synset('run.n.13') a row of unravelled stitches
Synset('discharge.n.06') the pouring forth of a fluid
```

```
Synset('run.n.15') an unbroken chronological sequence
Synset('run.n.16') a short trip
Synset('run.v.01') move fast by using one's feet, with one foot off the ground at any
Synset('scat.v.01') flee; take to one's heels; cut and run
Synset('run.v.03') stretch out over a distance, space, time, or scope; run or extend
Synset('operate.v.01') direct or control; projects, businesses, etc.
Synset('run.v.05') have a particular form
Synset('run.v.06') move along, of liquids
Synset('function.v.01') perform as expected when applied
Synset('range.v.01') change or be different within limits
Synset('campaign.v.01') run, stand, or compete for an office or a position
Synset('play.v.18') cause to emit recorded audio or video
Synset('run.v.11') move about freely and without restraint, or act as if running arou
Synset('tend.v.01') have a tendency or disposition to do or be something; be inclined
Synset('run.v.13') be operating, running or functioning
Synset('run.v.14') change from one state to another
Synset('run.v.15') cause to perform
Synset('run.v.16') be affected by; be subjected to
Synset('prevail.v.03') continue to exist
Synset('run.v.18') occur persistently
Synset('run.v.19') carry out a process or program, as on a computer or a machine
Synset('carry.v.15') include as the content; broadcast or publicize
Synset('run.v.21') carry out
Synset('guide.v.05') pass over, across, or through
Synset('run.v.23') cause something to pass or lead somewhere
Synset('run.v.24') make without a miss
Synset('run.v.25') deal in illegally, such as arms or liquor
Synset('run.v.26') cause an animal to move fast
Synset('run.v.27') be diffused
Synset('run.v.28') sail before the wind
Synset('run.v.29') cover by running; run a certain distance
Synset('run.v.30') extend or continue for a certain period of time
Synset('run.v.31') set animals loose to graze
Synset('run.v.32') keep company
Synset('run.v.33') run with the ball; in such sports as football
Synset('run.v.34') travel rapidly, by any (unspecified) means
Synset('ply.v.03') travel a route regularly
Synset('hunt.v.01') pursue for food or sport (as of wild animals)
Synset('race.v.02') compete in a race
Synset('move.v.13') progress by being changed
Synset('melt.v.01') reduce or cause to be reduced from a solid to a liquid state, usu
Synset('ladder.v.01') come unraveled or undone as if by snagging
Synset('run.v.41') become undone
```

```
sent = ['I', 'can', 'run', 'very', 'fast', '.']
print(lesk(sent, 'run', 'v'))
print(lesk(sent, 'run'))
```

```
Synset('scat.v.01')
Synset('scat.v.01')
```

The words that I chose shows a Wu-Palmer value low which means the similarity is pretty low. The two words sprint and run may be different in context when using both words similar for

The two words sprint and run may be different in context when using both words similar to dash.

The lesk gave scat as the output which means flee which gives a similar meaning to the sentence I inputted. This is pretty useful as it reads in a full sentence and outputs a word based on the sentence.

9.

SentiWordNet

SentiWordNet is a great method to analyze a text to get the deeper meaning as in the tone of the text whether it would be positive, neutral or negative. Though sometimes there may be some inaccuracies but overall a nice tool. Such as in text messages it is sometimes hard to predict how the text or on what context it has been written. It would be beneficial in classifying texts.

```
#9.
from nltk.corpus import sentiwordnet as swn # require to download package
senti_lst = list(swn.senti_synsets('hope'))
for x in senti_lst:
    print(x)
```

```
     <hope.n.01: PosScore=0.25 NegScore=0.125>
     <hope.n.02: PosScore=0.375 NegScore=0.0>
     <promise.n.02: PosScore=0.125 NegScore=0.25>
     <hope.n.04: PosScore=0.0 NegScore=0.375>
     <hope.n.05: PosScore=0.0 NegScore=0.0>
     <hope.n.06: PosScore=0.25 NegScore=0.0>
     <hope.v.01: PosScore=0.125 NegScore=0.125>
     <hope.v.02: PosScore=0.625 NegScore=0.25>
     <hope.v.03: PosScore=0.0 NegScore=0.0>
```

```
senti_lst = list(swn.senti_synsets('hope'))[0]
print("negative: ", senti_lst.neg_score())
print("positive: ", senti_lst.pos_score())
print("objective: ", senti_lst.obj_score())
```

```
     negative:  0.125
     positive:  0.25
     objective:  0.625
```

```
from nltk.corpus import sentiwordnet as swn

sent = 'We love and hate you!'
tokens = sent.split()

for token in tokens:
```

```
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        syn = syn_list[0]
        neg_score = syn.neg_score()
        pos_score = syn.pos_score()
        if pos_score > neg_score:
            print(f"{token}: Postive +({pos_score:.2f})")
        elif pos_score < neg_score:
            print(f"{token}: Negative -({neg_score:.2f})")
    else:
        print(f"{token}: Neutral ({pos_score:.2f})")
```

```
We: Neutral (0.12)
love: Postive +(0.62)
and: Neutral (0.62)
hate: Negative -(0.38)
you!: Neutral (0.12)
```

10.

Collocation

Collocation is when a word and another word may belong together in any case. A word that may be grouped together. It may not be always simple. Such as United States from text4.

```
import nltk
from nltk.book import *
text4
```

```
<Text: Inaugural Address Corpus>
```

```
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
```

```
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
```