CS4395_001_WordNet_AXP200075

February 26, 2023

Alan Perez | AXP200075 | CS 4395.001 | Assignment 3

0.0.1 Wordnet

is a lexical database of nouns, verbs, adjectives, and adverbs that provides short definitions called glosses, and use examples. Its grouped into cognitive synonymns (synsets)

[ ]:
```
import nltk
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn

nltk.download('stopwords')
nltk.download('treebank')
nltk.download('webtext')
nltk.download('nps_chat')
nltk.download('inaugural')
nltk.download('genesis')
nltk.download('gutenberg')
from nltk.book import *
text4
nltk.download('sentiwordnet')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package treebank to /root/nltk_data…
[nltk_data]   Package treebank is already up-to-date!
[nltk_data] Downloading package webtext to /root/nltk_data…
[nltk_data]   Package webtext is already up-to-date!
[nltk_data] Downloading package nps_chat to /root/nltk_data…
[nltk_data]   Package nps_chat is already up-to-date!
[nltk_data] Downloading package inaugural to /root/nltk_data…
[nltk_data]   Package inaugural is already up-to-date!
[nltk_data] Downloading package genesis to /root/nltk_data…
[nltk_data]   Package genesis is already up-to-date!
[nltk_data] Downloading package gutenberg to /root/nltk_data…
[nltk_data]   Package gutenberg is already up-to-date!
```

```
[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!
```

[ ]:
```
True
```

The code beloow gets all the synsets of 'game'

Each synset has an ID. The form of the ID is word.pos.nn

For example, 'game.n.01' is the first noun senset while 'game.v.01' is the first verb sysnet.

For a given synset, we can extract:

- a definition
- usage examples
- lemmas

[ ]:
```
wn.synsets("game")

print("Printing the synsets of game: ", wn.synsets("game"))
# hold synsets
syn = wn.synset("game.n.01")


# definition for the first noun synset
print("definition of game: ", wn.synset("game.n.01").definition())

# usage examples
print("usage examples of game.n.01: ", wn.synset("game.n.01").examples())

# lemmas
print("lemmas of game.n.01: ", wn.synset("game.n.01").lemmas())

game_synsets = wn.synsets('game', pos=wn.NOUN)
for syn in game_synsets:
  print("lemmas of game: ", syn.lemmas(), syn.definition())

# traversing the WordNet hierarchy
hyp = syn.hypernyms()[0]
top = wn.synset("entity.n.01")
while hyp:
  print("printing the hypernymns of noun: ", hyp)
  if hyp == top:
    break
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
```

```
Printing the synsets of game:  [Synset('game.n.01'), Synset('game.n.02'),
Synset('game.n.03'), Synset('game.n.04'), Synset('game.n.05'),
Synset('game.n.06'), Synset('game.n.07'), Synset('plot.n.01'),
Synset('game.n.09'), Synset('game.n.10'), Synset('game.n.11'),
Synset('bet_on.v.01'), Synset('crippled.s.01'), Synset('game.s.02')]
definition of game:  a contest with rules to determine a winner
usage examples of game.n.01:  ['you need four people to play this game']
lemmas of game.n.01:  [Lemma('game.n.01.game')]
lemmas of game:  [Lemma('game.n.01.game')] a contest with rules to determine a
winner
lemmas of game:  [Lemma('game.n.02.game')] a single play of a sport or other
contest
lemmas of game:  [Lemma('game.n.03.game')] an amusement or pastime
lemmas of game:  [Lemma('game.n.04.game')] animal hunted for food or sport
lemmas of game:  [Lemma('game.n.05.game')] (tennis) a division of play during
which one player serves
lemmas of game:  [Lemma('game.n.06.game')] (games) the score at a particular
point or the score needed to win
lemmas of game:  [Lemma('game.n.07.game')] the flesh of wild animals that is
used for food
lemmas of game:  [Lemma('plot.n.01.plot'), Lemma('plot.n.01.secret_plan'),
Lemma('plot.n.01.game')] a secret scheme to do something (especially something
underhand or illegal)
lemmas of game:  [Lemma('game.n.09.game')] the game equipment needed in order to
play a particular game
lemmas of game:  [Lemma('game.n.10.game'), Lemma('game.n.10.biz')] your
occupation or line of work
lemmas of game:  [Lemma('game.n.11.game')] frivolous or trifling behavior
printing the hypernymns of noun:  Synset('play.n.14')
printing the hypernymns of noun:  Synset('diversion.n.01')
printing the hypernymns of noun:  Synset('activity.n.01')
printing the hypernymns of noun:  Synset('act.n.02')
printing the hypernymns of noun:  Synset('event.n.01')
printing the hypernymns of noun:  Synset('psychological_feature.n.01')
printing the hypernymns of noun:  Synset('abstraction.n.06')
printing the hypernymns of noun:  Synset('entity.n.01')
```

WordNet is outputting the different meanings of 'game' and represents the different meanings along with the relationship to WordNet. I displayed each definition since of the synsets of game to show the different definitions 'game' could mean. The hierarchy of the synset has 'entity' as the top level and I noticed that as we went down the top level it started to become more specific which gives a better understanding of the words as we go down.

Output the following (or an empty list if none exist): hypernyms, hyponyms, meronyms, holonyms, antonym.

[ ]:
```
print("Hypernyms", syn.hypernyms())
print("Hynonymn", syn.hyponyms())
```

```
print("Meronyms", syn.member_meronyms())
print("Holonyms", syn.member_holonyms())
print("Antonym", syn.lemmas()[0].antonyms())
```

```
Hypernyms [Synset('play.n.14')]
Hynonymn []
Meronyms []
Holonyms []
Antonym []
```

1 Select a verb, Output all synsets

[ ]:
```
# select a verb
syn_verb = wn.synset("game.v.01")

# definition for the first verb synset
print("definition: ", wn.synset('game.v.01').definition())

# usage examples
print("usage examples: ", wn.synset('game.v.01').examples())

# lemmas
print("lemmas: ", wn.synset('game.v.01').lemmas())


# Printing the def of the synsets
verb_synsets = wn.synsets('game', pos=wn.VERB)

for luv in verb_synsets:
  lemmas = [l.name() for l in luv.lemmas()]
  print("Synset: " + luv.name() + "(" +luv.definition() + ") \n\t Lemmas: " +␣
 ↪str(lemmas))

# Traverse up the WordNet hierarchy
hyp = syn_verb.hypernyms()[0]
while hyp:
  print(hyp)
  if hyp.hypernyms():
    hyp = hyp.hypernyms()[0]
  else:
    break
```

```
definition:  place a bet on
usage examples:  ['Which horse are you backing?', "I'm betting on the new
horse"]
lemmas:  [Lemma('bet_on.v.01.bet_on'), Lemma('bet_on.v.01.back'),
Lemma('bet_on.v.01.gage'), Lemma('bet_on.v.01.stake'),
```

```
Lemma('bet_on.v.01.game'), Lemma('bet_on.v.01.punt')]
Synset: bet_on.v.01(place a bet on)
         Lemmas: ['bet_on', 'back', 'gage', 'stake', 'game', 'punt']
Synset('bet.v.02')
Synset('gamble.v.02')
Synset('play.v.01')
Synset('compete.v.01')
```

When traversing through hierarchy of a verb versus a non, I noticed that depending on the verb selected the meanings might be a bit broad. The use of a different verb and the related synsets could potentially give a better understand and the contexts of the verb.

2 Use morphy to find as many different forms of the word

[ ]:
```
print(wn.morphy("game", wn.NOUN))
print(wn.morphy("game", wn.VERB))
print(wn.morphy("game", wn.ADJ))
print(wn.morphy("play"))
```

```
game
game
game
play
```

3 Wu-Palmer & Lesk Algorithm

calculating the similarity of two words and checking the context

[ ]:
```
#two words that might be similar. Find the specific synsets you are interested
 ↪in.


sim1_synset = wn.synsets("fondness")
sim2_synset = wn.synsets("devotion")

print("First word: ", sim1_synset)
print("Second word: ", sim2_synset)

sim3_synset = wn.synsets("game")
sim4_synset = wn.synsets("play")

print("First word: ", sim1_synset)
print("Second word: ", sim2_synset)

print("Third word: ", sim3_synset)
print("Fourth word: ", sim4_synset)
```

```
First word:  [Synset('fondness.n.01'), Synset('affection.n.01'),
Synset('affectionateness.n.02')]
Second word:  [Synset('devotion.n.01'), Synset('devotion.n.02'),
Synset('idolatry.n.01'), Synset('devotion.n.04')]
First word:  [Synset('fondness.n.01'), Synset('affection.n.01'),
Synset('affectionateness.n.02')]
Second word:  [Synset('devotion.n.01'), Synset('devotion.n.02'),
Synset('idolatry.n.01'), Synset('devotion.n.04')]
Third word:  [Synset('game.n.01'), Synset('game.n.02'), Synset('game.n.03'),
Synset('game.n.04'), Synset('game.n.05'), Synset('game.n.06'),
Synset('game.n.07'), Synset('plot.n.01'), Synset('game.n.09'),
Synset('game.n.10'), Synset('game.n.11'), Synset('bet_on.v.01'),
Synset('crippled.s.01'), Synset('game.s.02')]
Fourth word:  [Synset('play.n.01'), Synset('play.n.02'), Synset('play.n.03'),
Synset('maneuver.n.03'), Synset('play.n.05'), Synset('play.n.06'),
Synset('bid.n.02'), Synset('play.n.08'), Synset('playing_period.n.01'),
Synset('free_rein.n.01'), Synset('shimmer.n.01'), Synset('fun.n.02'),
Synset('looseness.n.05'), Synset('play.n.14'), Synset('turn.n.03'),
Synset('gambling.n.01'), Synset('play.n.17'), Synset('play.v.01'),
Synset('play.v.02'), Synset('play.v.03'), Synset('act.v.03'),
Synset('play.v.05'), Synset('play.v.06'), Synset('play.v.07'),
Synset('act.v.05'), Synset('play.v.09'), Synset('play.v.10'),
Synset('play.v.11'), Synset('play.v.12'), Synset('play.v.13'),
Synset('play.v.14'), Synset('play.v.15'), Synset('play.v.16'),
Synset('play.v.17'), Synset('play.v.18'), Synset('toy.v.02'),
Synset('play.v.20'), Synset('dally.v.04'), Synset('play.v.22'),
Synset('dally.v.01'), Synset('play.v.24'), Synset('act.v.10'),
Synset('play.v.26'), Synset('bring.v.03'), Synset('play.v.28'),
Synset('play.v.29'), Synset('bet.v.02'), Synset('play.v.31'),
Synset('play.v.32'), Synset('play.v.33'), Synset('meet.v.10'),
Synset('play.v.35')]
```

[ ]:
```python
fond = wn.synset("fondness.n.01")
devote = wn.synset("devotion.n.01")

# initially comparring universe & universal, switched to game & play
univ1 = wn.synset("game.n.01")
univ2 = wn.synset("play.n.01")
# Run the Wu-Palmer similarity metric and the Lesk algorithm
# Wu-Palmer Similiarity Metric
print("similarity for fondness & devotion: ", wn.wup_similarity(fond, devote))

print("similarity for play and game: ", wn.wup_similarity(univ1, univ2))

# Lesk Algorithm, look at the definitions for 'game', 'play'
for ss in wn.synsets("devotion"):
  print(ss, ss.definition())
```

```
for ss in wn.synsets("game"):
  print(ss, ss.definition())

for ss in wn.synsets("play"):
  print(ss, ss.definition())

sent = ["I", "have", "the", "devotion", "for", "this", "game", "."]
sent2 = ["this", "game", "is", "fun", "to", "watch", "and", "play", "."]

# print(lesk(sent, "devotion", "n"))
print(lesk(sent, "devotion"))
print(lesk(sent2, "play"))
```

```
similarity for fondness & devotion:  0.6666666666666666
similarity for play and game:  0.2857142857142857
Synset('devotion.n.01') feelings of ardent love
Synset('devotion.n.02') commitment to some purpose
Synset('idolatry.n.01') religious zeal; the willingness to serve God
Synset('devotion.n.04') (usually plural) religious observance or prayers
(usually spoken silently)
Synset('game.n.01') a contest with rules to determine a winner
Synset('game.n.02') a single play of a sport or other contest
Synset('game.n.03') an amusement or pastime
Synset('game.n.04') animal hunted for food or sport
Synset('game.n.05') (tennis) a division of play during which one player serves
Synset('game.n.06') (games) the score at a particular point or the score needed
to win
Synset('game.n.07') the flesh of wild animals that is used for food
Synset('plot.n.01') a secret scheme to do something (especially something
underhand or illegal)
Synset('game.n.09') the game equipment needed in order to play a particular game
Synset('game.n.10') your occupation or line of work
Synset('game.n.11') frivolous or trifling behavior
Synset('bet_on.v.01') place a bet on
Synset('crippled.s.01') disabled in the feet or legs
Synset('game.s.02') willing to face danger
Synset('play.n.01') a dramatic work intended for performance by actors on a
stage
Synset('play.n.02') a theatrical performance of a drama
Synset('play.n.03') a preset plan of action in team sports
Synset('maneuver.n.03') a deliberate coordinated movement requiring dexterity
and skill
Synset('play.n.05') a state in which action is feasible
Synset('play.n.06') utilization or exercise
Synset('bid.n.02') an attempt to get something
Synset('play.n.08') activity by children that is guided more by imagination than
```

```
by fixed rules
Synset('playing_period.n.01') (in games or plays or other performances) the time
during which play proceeds
Synset('free_rein.n.01') the removal of constraints
Synset('shimmer.n.01') a weak and tremulous light
Synset('fun.n.02') verbal wit or mockery (often at another's expense but not to
be taken seriously)
Synset('looseness.n.05') movement or space for movement
Synset('play.n.14') gay or light-hearted recreational activity for diversion or
amusement
Synset('turn.n.03') (game) the activity of doing something in an agreed
succession
Synset('gambling.n.01') the act of playing for stakes in the hope of winning
(including the payment of a price for a chance to win a prize)
Synset('play.n.17') the act using a sword (or other weapon) vigorously and
skillfully
Synset('play.v.01') participate in games or sport
Synset('play.v.02') act or have an effect in a specified way or with a specific
effect or outcome
Synset('play.v.03') play on an instrument
Synset('act.v.03') play a role or part
Synset('play.v.05') be at play; be engaged in playful activity; amuse oneself in
a way characteristic of children
Synset('play.v.06') replay (as a melody)
Synset('play.v.07') perform music on (a musical instrument)
Synset('act.v.05') pretend to have certain qualities or state of mind
Synset('play.v.09') move or seem to move quickly, lightly, or irregularly
Synset('play.v.10') bet or wager (money)
Synset('play.v.11') engage in recreational activities rather than work; occupy
oneself in a diversion
Synset('play.v.12') pretend to be somebody in the framework of a game or playful
activity
Synset('play.v.13') emit recorded sound
Synset('play.v.14') perform on a certain location
Synset('play.v.15') put (a card or piece) into play during a game, or act
strategically as if in a card game
Synset('play.v.16') engage in an activity as if it were a game rather than take
it seriously
Synset('play.v.17') behave in a certain way
Synset('play.v.18') cause to emit recorded audio or video
Synset('toy.v.02') manipulate manually or in one's mind or imagination
Synset('play.v.20') use to one's advantage
Synset('dally.v.04') consider not very seriously
Synset('play.v.22') be received or accepted or interpreted in a specific way
Synset('dally.v.01') behave carelessly or indifferently
Synset('play.v.24') cause to move or operate freely within a bounded space
Synset('act.v.10') perform on a stage or theater
Synset('play.v.26') be performed or presented for public viewing
```

```
Synset('bring.v.03') cause to happen or to occur as a consequence
Synset('play.v.28') discharge or direct or be discharged or directed as if in a
continuous stream
Synset('play.v.29') make bets
Synset('bet.v.02') stake on the outcome of an issue
Synset('play.v.31') shoot or hit in a particular manner
Synset('play.v.32') use or move
Synset('play.v.33') employ in a game or in a specific position
Synset('meet.v.10') contend against an opponent in a sport, game, or battle
Synset('play.v.35') exhaust by allowing to pull on the line
Synset('idolatry.n.01')
Synset('play.v.15')
```

Based off my observations from the definition is that play can be a noun or verb there's more context and meaning that comes from the word play over devotionn, but when it came to the similarity for play and game it was rather low. I'm assuming the reason why the output was so low was due to how "play" and "game" could be seen as. When comparing "Devotion" and "Fondness" the similarity for the Wu Palmer algorithm it gave off a higher score.

When executing the Lesk algorithm it looks at the context of the words and compares them to check if its the correct meaning.

[ ]:
```
print(wn.synsets("distressed"))
swn_word = swn.senti_synsets("distressed")
# sent = swn.senti_synset("distressed")
print(swn_word)

# output the sentiment scores for each word
for x in swn_word:
  print(x)
  print(" Positive score = ", x.pos_score(), " Negative score = ", x.
 ↪neg_score(), " Objective score = ", x.obj_score())
```

```
[Synset('straiten.v.01'), Synset('distress.v.02'), Synset('distressed.s.01'),
Synset('dysphoric.a.01'), Synset('stressed.s.01'), Synset('disquieted.s.01')]
<filter object at 0x7f7826785880>
<straiten.v.01: PosScore=0.0 NegScore=0.625>
 Positive score =  0.0  Negative score =  0.625  Objective score =  0.375
<distress.v.02: PosScore=0.0 NegScore=0.5>
 Positive score =  0.0  Negative score =  0.5  Objective score =  0.5
<distressed.s.01: PosScore=0.0 NegScore=0.625>
 Positive score =  0.0  Negative score =  0.625  Objective score =  0.375
<dysphoric.a.01: PosScore=0.125 NegScore=0.875>
 Positive score =  0.125  Negative score =  0.875  Objective score =  0.0
<stressed.s.01: PosScore=0.125 NegScore=0.75>
 Positive score =  0.125  Negative score =  0.75  Objective score =  0.125
<disquieted.s.01: PosScore=0.0 NegScore=0.875>
 Positive score =  0.0  Negative score =  0.875  Objective score =  0.125
```

4 SentiWordNet - Polarity

SentiWordNet is a lexical resource thats built on top of WordNet, it showcases the positive, negative, and objective score for all possible synsets.

Selecting an emotionally charged word, finding their senti0synsets and outputting the polarity scores for each word.

[ ]:
```
# sentence for polarity
# sent = "From the beginning to the end Losers lose, winners win This is real,␣
 ↪we ain't got to pretend."

sent = "Sunny days wouldn't be special if it wasn't for rain, Joy wouldn't feel␣
 ↪so good if it wasn't for pain"
neg = 0
pos = 0

tokens = sent.split()
for token in tokens:
  syn_list = list(swn.senti_synsets(token))
  if syn_list:
    syn = syn_list[0]
    neg += syn.neg_score()
    pos += syn.pos_score()
    print("Word:", token, "pos = ", syn.pos_score(), "neg = ", syn.neg_score())

# pos and neg added together
print("neg\tpos counts")
print(neg, "\t", pos)
```

```
Word: Sunny pos =  0.5 neg =  0.0
Word: days pos =  0.0 neg =  0.0
Word: be pos =  0.0 neg =  0.0
Word: special pos =  0.0 neg =  0.0
Word: it pos =  0.0 neg =  0.0
Word: Joy pos =  0.5 neg =  0.25
Word: feel pos =  0.125 neg =  0.0
Word: so pos =  0.0 neg =  0.0
Word: good pos =  0.5 neg =  0.0
Word: it pos =  0.0 neg =  0.0
Word: pain pos =  0.0 neg =  0.75
neg     pos counts
1.0      1.625
```

Upon running this we can see the individual score for each word in our sentence. Towards the end each score is calculated and displayed. For possible use cases, Th9is could be used for software that checks what the user has typed so that it checks the emotion of the sentence. We could calculate the score of the sentence and check whether the sentence is positive or negative.

5 Collocation

Collocation is when two or more words usually occur together with frequency. They are words that go together for example “fast food”, “bright sun”, “deeply regret something”, etc each word has their own meaning but when put together it can create a new distinct meaning.

[ ]:
```
# get collactions

text4_collaction = text4.collocations()

print(text4_collaction)

text = ' '.join(text4.tokens)
text[:50]

import math

vocab = len(set(text4))
ow = text.count("Old World")/vocab
print("p(Old World) = ", ow)
o = text.count("Old")/vocab
print("p(Old) = ", o)

w = text.count("World")/vocab
print("p(World) = ", w)
pmi = math.log2(ow/ (o*w))



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(Old World) =  0.000997506234413965
p(Old) =  0.0010972568578553616
p(World) =  0.0017955112219451373
pmi =  8.983886091037398
```

We’re calculating the pointwise mutual information of “Old World” and checking the frequency the length of vocab and counting the number of times “Old World appears. We do this for”Old” and “World” as well and return the probability. The pmi score is 8.9 which indicates a strong association between the two words.