In [ ]:
```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('sentiwordnet')
nltk.download('gutenberg')
nltk.download('genesis')
nltk.download('inaugural')
nltk.download('nps_chat')
nltk.download('webtext')
nltk.download('treebank')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data] Downloading package sentiwordnet to /root/nltk_data...
[nltk_data]   Unzipping corpora/sentiwordnet.zip.
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Unzipping corpora/gutenberg.zip.
[nltk_data] Downloading package genesis to /root/nltk_data...
[nltk_data]   Unzipping corpora/genesis.zip.
[nltk_data] Downloading package inaugural to /root/nltk_data...
[nltk_data]   Unzipping corpora/inaugural.zip.
[nltk_data] Downloading package nps_chat to /root/nltk_data...
[nltk_data]   Unzipping corpora/nps_chat.zip.
[nltk_data] Downloading package webtext to /root/nltk_data...
[nltk_data]   Unzipping corpora/webtext.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[ ]:
```
True
```

Summary of Wordnet

Wordnet is a database of words. The database organizes nouns, adjectives, verbs, and adverbs. The database has the definition, antynonyms, hypernyms, meronyms, and many others that is related to the words.

Synset of Man (noun)

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

# Get synset of man
word.synsets('man')
```

Out[ ]:
```
[Synset('man.n.01'),
 Synset('serviceman.n.01'),
 Synset('man.n.03'),
 Synset('homo.n.02'),
 Synset('man.n.05'),
 Synset('man.n.06'),
 Synset('valet.n.01'),
 Synset('man.n.08'),
 Synset('man.n.09'),
 Synset('man.n.10'),
 Synset('world.n.08'),
 Synset('man.v.01'),
 Synset('man.v.02')]
```

In [ ]:
```
# Get definition of first synset of man
word.synset('man.n.01').definition()
```

Out[ ]:
```
'an adult person who is male (as opposed to a woman)'
```

In [ ]:
```
# Get example of first synset of man
word.synset('man.n.01').examples()
```

Out[ ]:
```
['there were two women and six men on the bus']
```

In [ ]:
```
# Get lemmas of first synset of man
word.synset('man.n.01').lemmas()
```

Out[ ]:
```
[Lemma('man.n.01.man'), Lemma('man.n.01.adult_male')]
```

In [ ]:
```
# Get hypernyms of man
hierarchy = word.synset('man.n.01').hypernyms()[0]

# Traversing through hierarchy and output it
while hierarchy:
  print(hierarchy)

  # Exit when encountering entity.n.01
  if(hierarchy == word.synset('entity.n.01')):
    break

  # Go to next hypernyms
  if(hierarchy.hypernyms()):
    hierarchy = hierarchy.hypernyms()[0]
```

```
Synset('adult.n.01')
Synset('person.n.01')
Synset('causal_agent.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')
```

In [ ]:
```
# Outputting hypernyms, hyponyms, meronyms, holonyms, antonym
print('Hypernyms: ', word.synset('man.n.01').hypernyms())
print('Root Hypernyms: ', word.synset('man.n.01').root_hypernyms())
print('Hyponyms: ', word.synset('man.n.01').hyponyms())
print('Part Meronyms: ', word.synset('man.n.01').part_meronyms())
print('Substance Meronyms: ', word.synset('man.n.01').substance_meronyms())
print('Part Holonyms: ', word.synset('man.n.01').part_holonyms())
print('Substance Holonyms: ', word.synset('man.n.01').substance_holonyms())
print("Antonym :", word.synset('man.n.01').lemmas()[0].antonyms())
```

```
Hypernyms:  [Synset('adult.n.01'), Synset('male.n.02')]
Root Hypernyms:  [Synset('entity.n.01')]
Hyponyms:  [Synset('adonis.n.01'), Synset('babu.n.01'), Synset('bachelor.n.
01'), Synset('bey.n.01'), Synset('black_man.n.01'), Synset('boy.n.02'), Syn
set('boyfriend.n.01'), Synset('bull.n.02'), Synset('dandy.n.01'), Synset('e
jaculator.n.01'), Synset('esquire.n.02'), Synset('eunuch.n.01'), Synset('ex
-boyfriend.n.01'), Synset('ex-husband.n.01'), Synset('father-figure.n.01'),
Synset('father_figure.n.01'), Synset('fellow.n.06'), Synset('galoot.n.01'),
Synset('geezer.n.01'), Synset('gentleman.n.01'), Synset('grass_widower.n.01
'), Synset('guy.n.01'), Synset('herr.n.01'), Synset('hooray_henry.n.01'), S
ynset('housefather.n.01'), Synset('hunk.n.01'), Synset('inamorato.n.01'), S
ynset('iron_man.n.01'), Synset('ironside.n.01'), Synset('middle-aged_man.n.
01'), Synset('monsieur.n.01'), Synset('old_boy.n.01'), Synset('old_man.n.01
'), Synset('patriarch.n.02'), Synset('peter_pan.n.01'), Synset('ponce.n.01
'), Synset('posseman.n.01'), Synset('senhor.n.01'), Synset('shaver.n.01'),
Synset('signor.n.01'), Synset('signore.n.01'), Synset('sir.n.01'), Synset('
stiff.n.01'), Synset('stud.n.01'), Synset('tarzan.n.01'), Synset('white_ma
n.n.01'), Synset('widower.n.01'), Synset('womanizer.n.01'), Synset('wonder_
boy.n.01'), Synset('yellow_man.n.01'), Synset('young_buck.n.01')]
Part Meronyms:  [Synset('adult_male_body.n.01')]
Substance Meronyms:  []
Part Holonyms:  []
Substance Holonyms:  []
Antonym : [Lemma('woman.n.01.woman')]
```

Reflection

I noticed that the WordNet organizes nouns based on if the nouns are similar to each other. Looking at the results when you move up the hierarchy, I noticed the synset of the nouns are similar to each other, much like a synonym. Thus, I see that the WordNet grouped nouns that are similar to each other.

Synset of Dance (verb)

In [ ]:
```
# Get synset of dance
word.synsets('dance')
```

Out[ ]:
```
[Synset('dance.n.01'),
 Synset('dance.n.02'),
 Synset('dancing.n.01'),
 Synset('dance.n.04'),
 Synset('dance.v.01'),
 Synset('dance.v.02'),
 Synset('dance.v.03')]
```

In [ ]:
```
# Get definition of first synset verb of dance
synset_dance = word.synset('dance.v.01')
synset_dance.definition()
```

Out[ ]:
```
'move in a graceful and rhythmical way'
```

In [ ]:
```
# Get example of first synset verb of dance
synset_dance.examples()
```

Out[ ]:
```
['The young girl danced into the room']
```

In [ ]:
```
# Get lemmas of first synset verb of dance
synset_dance.lemmas()
```

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

In [ ]:
```
# Get hypernyms o
hierarchy = synset_dance.hypernyms()[0]

# Traversing through hierarchy
while hierarchy:
  print(hierarchy)

  # Break if encounter move.v.03
  if(hierarchy == word.synset('move.v.03')):
    break

  # Go to the next hypernyms
  if(hierarchy.hypernyms()):
    hierarchy = hierarchy.hypernyms()[0]
```

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

In [ ]:
```
# Get hypernyms, hyponyms, meronyms, holonyms, and antonyms
print('Hypernyms: ', synset_dance.hypernyms())
print('Root Hypernyms: ', synset_dance.root_hypernyms())
print('Hyponyms: ', synset_dance.hyponyms())
print('Part Meronyms: ', synset_dance.part_meronyms())
print('Substance Meronyms: ', synset_dance.substance_meronyms())
print('Part Holonyms: ', synset_dance.part_holonyms())
print('Substance Holonyms: ', synset_dance.substance_holonyms())
print("Antonym :", synset_dance.lemmas()[0].antonyms())
```

```
Hypernyms:  [Synset('move.v.03')]
Root Hypernyms:  [Synset('move.v.03')]
Hyponyms:  [Synset('capriole.v.02'), Synset('chasse.v.01'), Synset('glissad
e.v.01')]
Part Meronyms:  []
Substance Meronyms:  []
Part Holonyms:  []
Substance Holonyms:  []
Antonym : []
```

Reflection

Much like the nouns, I noticed that the WordNet organizes verbs based on if they are similar to each other. I only see one synset when I traverse through the hierarchy with the verb dance. However, that one synset can be a synonym of dance. Thus, I see WordNet organizes verbs if they are similar to each other.

Using Morphy on Cried and Screamed

In [ ]:
```
# Using morphy on "cried" to see its root forms and output it
print(word.morphy('cried', word.VERB))
print(word.morphy('cried', word.NOUN))
print(word.morphy('cried', word.ADJ))

# Using morphy on "screamed" to see its root forms and output it
print(word.morphy('screamed', word.VERB))
print(word.morphy('screamed', word.NOUN))
print(word.morphy('screamed', word.ADJ))
```

```
cry
None
None
scream
None
None
```

In [ ]:
```
# Output synset of "cried" and "scream"
print(word.synsets('cried'))
print(word.synsets('scream'))
```

```
[Synset('shout.v.02'), Synset('cry.v.02'), Synset('exclaim.v.01'), Synset('
cry.v.04'), Synset('cry.v.05'), Synset('cry.v.06'), Synset('cry.v.07')]
[Synset('scream.n.01'), Synset('screech.n.01'), Synset('belly_laugh.n.02'),
Synset('shout.v.02'), Synset('yell.v.02'), Synset('scream.v.03')]
```

In [ ]:
```
# Outputting definition of cried and scream
print(word.synset('cry.v.02').definition())
print(word.synset('scream.v.01').definition())
```

```
shed tears because of sadness, rage, or pain
utter a sudden loud cry
```

In [ ]:
```
cried = word.synset('cry.v.02')
scream = word.synset('scream.v.01')

# Use Wu-Palmer similarity metric to see if the two words are similar
word.wup_similarity(cried, scream)
```

Out[ ]:
```
0.3333333333333333
```

In [ ]:
```
from nltk.wsd import lesk

sentence = ['I', 'cried', 'tears', 'when', 'my', 'dog', 'died', '.']
sentence_two = ['I', 'scream', 'when', 'I', 'saw', 'the', 'bug', '.']

# Use the lesk algorithm to see the "cried" synset in the first sentence an
d
# "scream" synset in the second sentence
print(lesk(sentence, 'cried'))
print(lesk(sentence_two, 'scream'))
```

```
Synset('cry.v.02')
Synset('yell.v.02')
```

Reflection

I used "cried" (synset('cry.v.02')) on the first sentence, and noticed that the lesk algorithm got the correct synset. However, on the second sentence, I used "scream" (synset('scream.v.01')) and the lesk algorithm got it wrong. Therefore, I see that the algorithm is not perfect, and that it will make mistakes.

I thought that "cried" and "scream" was very similar to each other. However, I noticed that the Wu-Palmer similarity metric says otherwise. The metric shows 0.3333, which would mean that the two words are not that similar to each other.

SentiWordNet

SentiWordNet has a functionality that lets you break down the words in the sentence and see the positive and negative score of the word. The positive and negative score can help someone see the overall tone of the sentence or the tone of the word. Many people can use SentiWordNet to see how they can respond to something if they are unsure of the tone of the sentence is positive or negative. Also, many article writers can use SentiWordNet to find words to evoke emotion to the readers.

In [ ]:
```
from nltk.corpus import sentiwordnet as senti

# Get senti synset of despair
senti_synset = list(senti.senti_synsets('despair'))

# Output results
for synset in senti_synset:
  print(synset)
```

```
<despair.n.01: PosScore=0.0 NegScore=0.25>
<despair.n.02: PosScore=0.25 NegScore=0.625>
<despair.v.01: PosScore=0.0 NegScore=0.0>
```

In [ ]:

```
# Create sentence
sentence_three = 'Mark cried tears when he found his lost cat.'
split_sentence = sentence_three.split()

# Loop to go through the words in sentence and find the senti synset for th
e words
for tokens in split_sentence:
  senti_list = list(senti.senti_synsets(tokens))

  # Loop to output results to screen
  for synset in senti_list:
    print(synset)
```

<mark.n.01: PosScore=0.125 NegScore=0.125>
<marker.n.02: PosScore=0.0 NegScore=0.0>
<target.n.01: PosScore=0.0 NegScore=0.0>
<mark.n.04: PosScore=0.0 NegScore=0.0>
<mark.n.05: PosScore=0.0 NegScore=0.0>
<mark.n.06: PosScore=0.0 NegScore=0.0>
<mark.n.07: PosScore=0.0 NegScore=0.0>
<mark.n.08: PosScore=0.0 NegScore=0.0>
<chump.n.01: PosScore=0.125 NegScore=0.125>
<mark.n.10: PosScore=0.0 NegScore=0.0>
<sign.n.01: PosScore=0.0 NegScore=0.0>
<mark.n.12: PosScore=0.0 NegScore=0.0>
<scratch.n.10: PosScore=0.0 NegScore=0.25>
<crisscross.n.01: PosScore=0.0 NegScore=0.0>
<bell_ringer.n.03: PosScore=0.0 NegScore=0.0>
<tag.v.01: PosScore=0.0 NegScore=0.0>
<mark.v.02: PosScore=0.0 NegScore=0.0>
<distinguish.v.03: PosScore=0.0 NegScore=0.0>
<commemorate.v.01: PosScore=0.0 NegScore=0.0>
<mark.v.05: PosScore=0.0 NegScore=0.0>
<stigmatize.v.01: PosScore=0.0 NegScore=0.0>
<notice.v.02: PosScore=0.0 NegScore=0.0>
<scar.v.01: PosScore=0.0 NegScore=0.0>
<score.v.02: PosScore=0.0 NegScore=0.0>
<set.v.04: PosScore=0.5 NegScore=0.0>
<score.v.03: PosScore=0.0 NegScore=0.0>
<cross_off.v.01: PosScore=0.0 NegScore=0.0>
<check.v.06: PosScore=0.0 NegScore=0.0>
<grade.v.03: PosScore=0.375 NegScore=0.0>
<punctuate.v.01: PosScore=0.0 NegScore=0.0>
<shout.v.02: PosScore=0.0 NegScore=0.0>
<cry.v.02: PosScore=0.0 NegScore=0.75>
<exclaim.v.01: PosScore=0.125 NegScore=0.5>
<cry.v.04: PosScore=0.0 NegScore=0.0>
<cry.v.05: PosScore=0.0 NegScore=0.0>
<cry.v.06: PosScore=0.0 NegScore=0.0>
<cry.v.07: PosScore=0.0 NegScore=0.0>
<crying.n.01: PosScore=0.0 NegScore=0.0>
<tear.n.01: PosScore=0.0 NegScore=0.0>
<rip.n.02: PosScore=0.0 NegScore=0.0>
<bust.n.04: PosScore=0.0 NegScore=0.25>
<tear.n.04: PosScore=0.0 NegScore=0.0>
<tear.v.01: PosScore=0.0 NegScore=0.0>
<tear.v.02: PosScore=0.0 NegScore=0.0>
<tear.v.03: PosScore=0.0 NegScore=0.0>
<pluck.v.05: PosScore=0.0 NegScore=0.0>
<tear.v.05: PosScore=0.0 NegScore=0.0>
<helium.n.01: PosScore=0.0 NegScore=0.0>
<he.n.02: PosScore=0.0 NegScore=0.0>
<found.n.01: PosScore=0.0 NegScore=0.0>
<establish.v.01: PosScore=0.0 NegScore=0.0>
<establish.v.02: PosScore=0.0 NegScore=0.0>
<establish.v.08: PosScore=0.0 NegScore=0.0>
<find.v.01: PosScore=0.0 NegScore=0.0>
<detect.v.01: PosScore=0.0 NegScore=0.0>
<find.v.03: PosScore=0.0 NegScore=0.0>

```
<determine.v.01: PosScore=0.0 NegScore=0.0>
<find.v.05: PosScore=0.125 NegScore=0.0>
<witness.v.02: PosScore=0.125 NegScore=0.0>
<line_up.v.02: PosScore=0.0 NegScore=0.0>
<discover.v.03: PosScore=0.0 NegScore=0.0>
<discover.v.04: PosScore=0.0 NegScore=0.0>
<find.v.10: PosScore=0.0 NegScore=0.125>
<rule.v.04: PosScore=0.0 NegScore=0.0>
<receive.v.02: PosScore=0.0 NegScore=0.0>
<find.v.13: PosScore=0.0 NegScore=0.0>
<recover.v.01: PosScore=0.0 NegScore=0.0>
<find.v.15: PosScore=0.0 NegScore=0.0>
<find_oneself.v.01: PosScore=0.0 NegScore=0.0>
<found.a.01: PosScore=0.0 NegScore=0.0>
<doomed.n.01: PosScore=0.0 NegScore=0.0>
<lose.v.01: PosScore=0.0 NegScore=0.5>
<lose.v.02: PosScore=0.0 NegScore=0.5>
<lose.v.03: PosScore=0.0 NegScore=0.5>
<misplace.v.01: PosScore=0.0 NegScore=0.125>
<lose.v.05: PosScore=0.0 NegScore=0.125>
<lose.v.06: PosScore=0.0 NegScore=0.0>
<lose.v.07: PosScore=0.0 NegScore=0.125>
<lose.v.08: PosScore=0.0 NegScore=0.125>
<fall_back.v.04: PosScore=0.0 NegScore=0.0>
<miss.v.01: PosScore=0.0 NegScore=0.25>
<suffer.v.11: PosScore=0.0 NegScore=0.25>
<lost.a.01: PosScore=0.0 NegScore=0.75>
<confused.s.03: PosScore=0.0 NegScore=0.0>
<lost.a.03: PosScore=0.0 NegScore=0.625>
<lost.a.04: PosScore=0.0 NegScore=0.625>
<lost.s.05: PosScore=0.0 NegScore=0.5>
<lost.s.06: PosScore=0.125 NegScore=0.625>
<bemused.s.01: PosScore=0.125 NegScore=0.0>
<baffled.s.01: PosScore=0.0 NegScore=0.5>
<helpless.s.02: PosScore=0.0 NegScore=0.875>
```

Reflection

I observed that most of the negative scores the words have has scores more than 0.000. Furthermore, I observed that most of the positive scores the words have scores of 0.000. Thus, I can conclude that the sentence I made up have a negative tone and the words have a negative sentiment. I can use the the scores in an NLP application, specifically chatbot. With the scores, I can make the chatbot write back an appropiate response. So if a user writes a sentence with a negative score, the chatbot can ask if something is wrong and if the chatbot can help.

Collocation

Collocation is when two or more words are utilized together in a sentence to show something or describe something. An example of a collocation is American people. American people is a collocation that is used to describe the people living in America.

In [ ]:
```
from nltk.book import text4
import math

# Get collocations from text4 and output it
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
```