# Portfolio Assignment: WordNet

## 1

WordNet is a hierarchal organization of different parts of speech. The purpose of WordNet is to reflect how people organize words in their own minds in the form of a database. Common relationships between words in WordNet include synonymy, hyperonymy, etc)

## 2

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

wn.synsets('trunk')
```

Out[1]:
```
[Synset('trunk.n.01'),
 Synset('trunk.n.02'),
 Synset('torso.n.01'),
 Synset('luggage_compartment.n.01'),
 Synset('proboscis.n.02')]
```

## 3

In [2]:
```
print('Definitions:', wn.synset('luggage_compartment.n.01').definition(),'\n')
print('Usage Examples:', wn.synset('luggage_compartment.n.01').examples(),'\n')
print('Lemmas:', wn.synset('luggage_compartment.n.01').lemmas(),'\n')
```

```
Definitions: compartment in an automobile that carries luggage or shopping or tools

Usage Examples: ['he put his golf bag in the trunk']

Lemmas: [Lemma('luggage_compartment.n.01.luggage_compartment'), Lemma('luggage_compartme
nt.n.01.automobile_trunk'), Lemma('luggage_compartment.n.01.trunk')]
```

In [3]:
```
trunk = wn.synset('luggage_compartment.n.01')
hyper = lambda s: s.hypernyms()
list(trunk.closure(hyper))
```

Out[3]:
```
[Synset('compartment.n.02'),
 Synset('room.n.01'),
 Synset('area.n.05'),
 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')]
```

For nouns, the WordNet hierarchy begins with entity, goes down to the type of entity, then continues to go down until it reaches the actual noun itself.

## 4

In [4]:
```
print('Hypernyms:', wn.synset('luggage_compartment.n.01').hypernyms())
print('Hyponyms:', wn.synset('luggage_compartment.n.01').hyponyms())
print('Meronyms:', wn.synset('luggage_compartment.n.01').part_meronyms())
print('Holonyms:', wn.synset('luggage_compartment.n.01').part_holonyms())
print('Antonyms:', [l.antonyms() for l in wn.synset('luggage_compartment.n.01').lemmas(
```

```
Hypernyms: [Synset('compartment.n.02')]
Hyponyms: [Synset('boot.n.02')]
Meronyms: []
Holonyms: [Synset('car.n.01')]
Antonyms: [[], [], []]
```

## 5

In [5]:
```
wn.synsets('lie')
```

Out[5]:
```
[Synset('lie.n.01'),
 Synset('lie.n.02'),
 Synset('lie.n.03'),
 Synset('lie.v.01'),
 Synset('lie.v.02'),
 Synset('dwell.v.02'),
 Synset('lie.v.04'),
 Synset('lie.v.05'),
 Synset('lie.v.06'),
 Synset('lie_down.v.01')]
```

## 6

In [6]:
```
print('Definitions:', wn.synset('lie.v.05').definition(),'\n')
print('Usage Examples:', wn.synset('lie.v.05').examples(),'\n')
print('Lemmas:', wn.synset('lie.v.05').lemmas(),'\n')
```

```
Definitions: tell an untruth; pretend with intent to deceive

Usage Examples: ["Don't lie to your parents", 'She lied when she told me she was only 2
9']

Lemmas: [Lemma('lie.v.05.lie')]
```

In [7]:
```
trunk = wn.synset('lie.v.05')
hyper = lambda s: s.hypernyms()
list(trunk.closure(hyper))
```

Out[7]:
```
[Synset('misinform.v.01'),
 Synset('inform.v.01'),
 Synset('communicate.v.02'),
 Synset('interact.v.01'),
 Synset('act.v.01')]
```

Unlike nouns, there is no top-level synset for verbs. Instead, the hierarchy continues up only until

the high possible level that it can.

## 7

In [8]:
```
wn.morphy('lie', wn.VERB)
```

Out[8]:
```
'lie'
```

## 8

In [9]:
```
from nltk.wsd import lesk

cougar = wn.synset('cougar.n.01')
panther = wn.synset('panther.n.02')

print('Wu-Palmer Similarity:', wn.wup_similarity(cougar, panther))

sent = ['Cougars are smaller than lions, but are still dangerous']
print('Lesk:', lesk(sent, 'panther'))
```

```
Wu-Palmer Similarity: 0.8125
Lesk: Synset('panther.n.02')
```

The Wu-Palmer Similarity metric and the Lesk algorithm are both effective at capturing the similarity of words.

## 9

SentiWord is a resource built for sentiment analysis. The way it works is that it assigns positive, negative, and objective scores to synsets from WordNet that can be used to to understand someone's opinion.

In [10]:
```
import nltk
nltk.download('sentiwordnet')
from nltk.corpus import sentiwordnet as swn

hate_list = swn.senti_synsets('hate')
for hate in hate_list:
    print(hate)
    print('Positive Score:', hate.pos_score())
    print('Negative Score:', hate.neg_score())
    print('Objective Score:', hate.obj_score())
    print('\n')
```

```
[nltk_data] Downloading package sentiwordnet to
[nltk_data]     C:\Users\iront\AppData\Roaming\nltk_data...
[nltk_data]   Package sentiwordnet is already up-to-date!
<hate.n.01: PosScore=0.125 NegScore=0.375>
Positive Score: 0.125
Negative Score: 0.375
Objective Score: 0.5
```

```
<hate.v.01: PosScore=0.0 NegScore=0.75>
Positive Score: 0.0
Negative Score: 0.75
Objective Score: 0.25
```

In [11]:

```
sent = 'I hate Indiana Jones'
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if syn_list:
        print(syn_list[0])
        print('Positive Score:', syn_list[0].pos_score())
        print('Negative Score:', syn_list[0].neg_score())
        print('Objective Score:', syn_list[0].obj_score())
        print('\n')
```

```
<iodine.n.01: PosScore=0.0 NegScore=0.0>
Positive Score: 0.0
Negative Score: 0.0
Objective Score: 1.0


<hate.n.01: PosScore=0.125 NegScore=0.375>
Positive Score: 0.125
Negative Score: 0.375
Objective Score: 0.5


<indiana.n.01: PosScore=0.0 NegScore=0.0>
Positive Score: 0.0
Negative Score: 0.0
Objective Score: 1.0


<jones.n.01: PosScore=0.0 NegScore=0.0>
Positive Score: 0.0
Negative Score: 0.0
Objective Score: 1.0
```

WordNet assigns scores only to synsets that are commonly used in opinionated ways; otherwise, default scores are used. One application for these scores is summing up all the scores in a sentence in order to understand the overall polarity of the sentence.

## 10

A collocation is a set of words that are commonly found together, despite sometimes having very different meanings. For example "right" and "now" mean very different things, but "right now" is commonly said.

In [12]:

```
from nltk.book import *

text4.collocations()
```

```
*** 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
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
```

In [13]:

```
text = ' '.join(text4.tokens)

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

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

While simple, the formula for point-wise mutual information is very effective and can tell us which words are likely to be collocations.