In [382…

```
from nltk.corpus import wordnet as wn
from nltk.wsd import lesk
from nltk.corpus import sentiwordnet as swn
from nltk.book import *
import math
```

1. Wordnet is a NLTK package. It is a word database that includes words synonyms groups as synsets. It can be used to find/group words of similar meaning.

1. Code written below.

In [383…

```
wn.synsets('chair')
```

Out[383…

```
[Synset('chair.n.01'),
 Synset('professorship.n.01'),
 Synset('president.n.04'),
 Synset('electric_chair.n.01'),
 Synset('chair.n.05'),
 Synset('chair.v.01'),
 Synset('moderate.v.01')]
```

1. Code written below. WordNet nouns are organized from basic (entity.n.01) to most specific (in this case, chair.n.o1). There are many synsets between entity and chair, each getting more specific as it goes.

In [384…

```
print(wn.synset('chair.n.01').definition()) #definition
print(wn.synset('chair.n.01').examples()) #examples
print(wn.synset('chair.n.01').lemmas()) #lemmas
path = wn.synset('chair.n.01').hypernym_paths()[0] #hierarchy synsets
for element in path:
    print(element)
```

```
a seat for one person, with a support for the back
['he put his coat over the back of the chair and sat down']
[Lemma('chair.n.01.chair')]
Synset('entity.n.01')
Synset('physical_entity.n.01')
Synset('object.n.01')
Synset('whole.n.02')
Synset('artifact.n.01')
Synset('instrumentality.n.03')
Synset('furnishing.n.02')
Synset('furniture.n.01')
Synset('seat.n.03')
Synset('chair.n.01')
```

1. Code written below.

In [385…

```
print(wn.synset('chair.n.01').hypernyms())
print(wn.synset('chair.n.01').hyponyms())
print(wn.synset('chair.n.01').part_meronyms())
print(wn.synset('chair.n.01').part_holonyms())
print(wn.synset('chair.n.01').lemmas()[0].antonyms())
```

```
[Synset('seat.n.03')]
[Synset('armchair.n.01'), Synset('barber_chair.n.01'), Synset('chair_of_state.n.01'), Synset('chaise_longue.n.01'), Synset('eames_chair.n.01'), Synset('fighting_chair.n.01'), Synset('folding_chair.n.01'), Synset('highchair.n.01'), Synset('ladder-back.n.01'), Synset('lawn_chair.n.01'), Synset('rocking_chair.n.01'), Synset('straight_chair.n.01'), Synset('swivel_chair.n.01'), Synset('tablet-armed_chair.n.01'), Synset('wheelchair.n.01')]
[Synset('back.n.08'), Synset('leg.n.03')]
[]
[]
```

1. Code written below.

In [386…

```
wn.synsets('throw')
```

Out[386…

```
[Synset('throw.n.01'),
 Synset('throw.n.02'),
 Synset('throw.n.03'),
 Synset('throw.n.04'),
 Synset('throw.n.05'),
 Synset('throw.v.01'),
 Synset('throw.v.02'),
 Synset('shed.v.01'),
 Synset('throw.v.04'),
 Synset('give.v.07'),
 Synset('throw.v.06'),
 Synset('project.v.10'),
 Synset('throw.v.08'),
 Synset('bewilder.v.02'),
 Synset('hurl.v.03'),
 Synset('hold.v.03'),
 Synset('throw.v.12'),
 Synset('throw.v.13'),
 Synset('throw.v.14'),
 Synset('confuse.v.02')]
```

1. Code written below. Similar to nouns, verbs begin with entity, and becomes more specific as the synsets go. Compared to nouns like 'object', verbs seem to be more abstract like 'abstraction', 'psychological_feature', etc.

In [387…

```
print(wn.synset('throw.n.01').definition()) #definition
print(wn.synset('throw.n.01').examples()) #examples
print(wn.synset('throw.n.01').lemmas()) #lemmas
path = wn.synset('throw.n.01').hypernym_paths()[0] #hierarchy synsets
for element in path:
    print(element)
```

```
the act of throwing (propelling something with a rapid movement of the arm and wrist)
['the catcher made a good throw to second base']
[Lemma('throw.n.01.throw')]
Synset('entity.n.01')
Synset('abstraction.n.06')
Synset('psychological_feature.n.01')
Synset('event.n.01')
Synset('act.n.02')
Synset('propulsion.n.02')
Synset('throw.n.01')
```

1. Code written below. Not sure how to go about this, just did different forms of the word to show that root is the same.

In [388…

```
print(wn.morphy('throw', wn.VERB))
print(wn.morphy('throwing', wn.VERB))
print(wn.morphy('threw', wn.VERB))
```

```
throw
throw
throw
```

1. Code written below. The similarity wasn't that high, only 59%. The lesk algorithm shows that the correct synsets were not the ones used for Wu-Palmer, and the synsets for 'Toss' became 'Flip'.

In [389…

```
print(wn.synset('throw.n.01').wup_similarity(wn.synset('toss.n.01'))) #wu-palmer for throw and toss
print(lesk('Throw the football', 'Throw'))
print(lesk('Toss the football', 'Toss'))
```

```
0.5882352941176471
Synset('throw.v.14')
Synset('flip.v.06')
```

1. Code written below. SentiWordNet is used for finding polarity of words, with the intention of finding opinion of the words. It can be used to determine opinion of a letter or message, which is sometimes hard to determine from text on its own. For the sentence, I would say NLTK did an okay job of identifying and scoring words, the utility of knowing the polarity for an NLP app is seeing how well the application can truly identify the meaning of the speech.

In [390…

```
hatred = swn.senti_synset('hatred.n.01')
print(hatred)
print("Hatred positive score = ", hatred.pos_score())
print("Hatred negative score = ", hatred.neg_score())
print("Hatred objective score = ", hatred.obj_score())

sent = 'I hated the burger but the fries were amazing.'
tokens = sent.split()
for token in tokens:
    syn_list = list(swn.senti_synsets(token))
    if len(syn_list) > 0:
        syn = syn_list[0]
        print(str(syn_list[0]) + "negative: ", syn.neg_score())
        print(str(syn_list[0]) + "positive: ", syn.pos_score())
        print(str(syn_list[0]) + "objective: ", syn.obj_score())
```

```
<hate.n.01: PosScore=0.125 NegScore=0.375>
Hatred positive score =  0.125
Hatred negative score =  0.375
Hatred objective score =  0.5
<iodine.n.01: PosScore=0.0 NegScore=0.0>negative:  0.0
<iodine.n.01: PosScore=0.0 NegScore=0.0>positive:  0.0
<iodine.n.01: PosScore=0.0 NegScore=0.0>objective:  1.0
<hate.v.01: PosScore=0.0 NegScore=0.75>negative:  0.75
<hate.v.01: PosScore=0.0 NegScore=0.75>positive:  0.0
<hate.v.01: PosScore=0.0 NegScore=0.75>objective:  0.25
<burger.n.01: PosScore=0.0 NegScore=0.0>negative:  0.0
<burger.n.01: PosScore=0.0 NegScore=0.0>positive:  0.0
<burger.n.01: PosScore=0.0 NegScore=0.0>objective:  1.0
<merely.r.01: PosScore=0.0 NegScore=0.0>negative:  0.0
<merely.r.01: PosScore=0.0 NegScore=0.0>positive:  0.0
<merely.r.01: PosScore=0.0 NegScore=0.0>objective:  1.0
<french_fries.n.01: PosScore=0.125 NegScore=0.0>negative:  0.0
<french_fries.n.01: PosScore=0.125 NegScore=0.0>positive:  0.125
<french_fries.n.01: PosScore=0.125 NegScore=0.0>objective:  0.875
<be.v.01: PosScore=0.25 NegScore=0.125>negative:  0.125
<be.v.01: PosScore=0.25 NegScore=0.125>positive:  0.25
<be.v.01: PosScore=0.25 NegScore=0.125>objective:  0.625
```

1. Code written below. A collocation is a collection of words who create a meaning that is more than the meaning of the individual words themselves. An example of this is 'hot dog'. The results of the mutual information show a pmi of 3.87, which is pretty high, showing that Federal Government can be considered a collocation.

In [391…

```
print(text4.collocations()) #prints collocations

text = ' '.join(text4.tokens)
vocab = len(set(text4))
fg = text.count('Federal Government')/vocab
print("p(Federal Government) = ", fg)
f = text.count('Federal')/vocab
print("p(Federal) = ", f)
g = text.count('Government')/vocab
print('p(Government) = ', g)
pmi = math.log2(fg / (f * g))
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(Federal Government) =  0.0031920199501246885
p(Federal) =  0.006483790523690773
p(Government) =  0.033715710723192202
pmi =  3.868067366919006
```