

▼ Part 0 - Basic Setup

```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('sentiwordnet')
nltk.download('book')
```

```
↳ [nltk_data] | Unzipping corpora/ppattach.zip.
[nltk_data] | Downloading package reuters to /root/nltk_data...
[nltk_data] | Downloading package senseval to /root/nltk_data...
[nltk_data] | Unzipping corpora/senseval.zip.
[nltk_data] | Downloading package state_union to /root/nltk_data...
[nltk_data] | Unzipping corpora/state_union.zip.
[nltk_data] | Downloading package stopwords to /root/nltk_data...
[nltk_data] | Unzipping corpora/stopwords.zip.
[nltk_data] | Downloading package swadesh to /root/nltk_data...
[nltk_data] | Unzipping corpora/swadesh.zip.
[nltk_data] | Downloading package timit to /root/nltk_data...
[nltk_data] | Unzipping corpora/timit.zip.
[nltk_data] | Downloading package treebank to /root/nltk_data...
[nltk_data] | Unzipping corpora/treebank.zip.
[nltk_data] | Downloading package toolbox to /root/nltk_data...
[nltk_data] | Unzipping corpora/toolbox.zip.
[nltk_data] | Downloading package udhr to /root/nltk_data...
[nltk_data] | Unzipping corpora/udhr.zip.
[nltk_data] | Downloading package udhr2 to /root/nltk_data...
[nltk_data] | Unzipping corpora/udhr2.zip.
[nltk_data] | Downloading package unicode_samples to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/unicode_samples.zip.
[nltk_data] | Downloading package webtext to /root/nltk_data...
[nltk_data] | Unzipping corpora/webtext.zip.
[nltk_data] | Downloading package wordnet to /root/nltk_data...
[nltk_data] | Package wordnet is already up-to-date!
[nltk_data] | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data] | Unzipping corpora/wordnet_ic.zip.
[nltk_data] | Downloading package words to /root/nltk_data...
[nltk_data] | Unzipping corpora/words.zip.
[nltk_data] | Downloading package maxent_treebank_pos_tagger to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping taggers/maxent_treebank_pos_tagger.zip.
[nltk_data] | Downloading package maxent_ne_chunker to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] | Downloading package universal_tagset to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping taggers/universal_tagset.zip.
[nltk_data] | Downloading package punkt to /root/nltk_data...
[nltk_data] | Package punkt is already up-to-date!
[nltk_data] | Downloading package book_grammars to
[nltk_data] | /root/nltk_data...
```

```
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping grammars/book_grammars.zip.
[nltk_data] | Downloading package city_database to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping corpora/city_database.zip.
[nltk_data] | Downloading package tagsets to /root/nltk_data...
[nltk_data] | Unzipping help/tagsets.zip.
[nltk_data] | Downloading package panlex_swadesh to
[nltk_data] | /root/nltk_data...
[nltk_data] | Downloading package averaged_perceptron_tagger to
[nltk_data] | /root/nltk_data...
[nltk_data] | Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] |
[nltk_data] | Done downloading collection book
True
```

```

print(synset.definition())
print(synset.examples())
print(synset.lemmas())

(astronomy) a celestial body of hot gases that radiates energy derived from thermonuclear fusion
[]
[Lemma('star.n.01.star')]

```

```

hyper = synset.hypernyms()[0]
while hyper:
    print(hyper)
    hyper = hyper.hypernyms()
    if hyper:
        hyper = hyper[0]

Synset('celestial_body.n.01')
Synset('natural_object.n.01')
Synset('whole.n.02')
Synset('object.n.01')
Synset('physical_entity.n.01')
Synset('entity.n.01')

```

I noticed that for nouns WordNet is organized in a hierarchical manner. At the very top levels all nouns are entities, and a star is a physical object. Perhaps a non-physical noun would be something like anger.

▼ Part 4

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

```

print(synset.hypernyms())
print(synset.hyponyms())
print(synset.part_meronyms())
print(synset.part_holonyms())
print(synset.lemmas()[0].antonyms())

[Synset('celestial_body.n.01')]
[Synset('binary_star.n.01'), Synset('fixed_star.n.01'), Synset('giant_star.n.01'), Synset('main_sequence_star.n.01'), Synset('nova.n.01'), Synset('supernova.n.01')]
[]
[]
[]

```

▼ Part 5

Select a verb. Output all synsets.

```
verb = 'condense'
v_synsets = wn.synsets(verb)
print(v_synsets)
```

```
[Synset('condense.v.01'), Synset('digest.v.07'), Synset('condense.v.03'), Synset('conder
```



▼ Part 6

Select one synset from the list of synsets. Extract its definition, usage examples, and lemmas. From your selected synset, traverse up the WordNet hierarchy as far as you can, outputting the synsets as you go. Write a couple of sentences observing the way that WordNet is organized for verbs.

```
v_synset = v_synsets[0]
print(v_synset.definition())
print(v_synset.examples())
print(v_synset.lemmas())
```

```
undergo condensation; change from a gaseous to a liquid state and fall in drops
['water condenses', 'The acid distills at a specific temperature']
[Lemma('condense.v.01.condense'), Lemma('condense.v.01.distill'), Lemma('condense.v.01.c
```



```
hyper = v_synset.hypernyms()[0]
while hyper:
    print(hyper)
    hyper = hyper.hypernyms()
    if hyper:
        hyper = hyper[0]

    Synset('liquefy.v.03')
    Synset('change_integrity.v.01')
    Synset('change.v.02')
```

I noticed that there is not top level analogue for 'entity' (for noun synsets) in verb synsets. As such, since verbs are harder to organize there seem to be fewer hierarchical levels.

▼ Part 7

Use morphy to find as many different forms of the word as you can.

```
print(wn.morphy('condensing'))
print(wn.morphy('condensed'))

condensing
condense
```

▼ Part 8

Select two words that you think might be similar. Find the specific synsets you are interested in. Run the Wu-Palmer similarity metric and the Lesk algorithm. Write a couple of sentences with your observations.

```
from nltk.wsd import lesk
from nltk import word_tokenize

word1, word2 = 'photon.n.01', 'electron.n.01'
syn1, syn2 = wn.synset(word1), wn.synset(word2)
print('Wu-Palmer:', wn.wup_similarity(syn1, syn2))

sent = 'A photon was used in the dual-slit experiment, uncovering the basics of quantum mecha
tokens = word_tokenize(sent)
print(lesk(tokens, 'photon'))

Wu-Palmer: 0.7
Synset('photon.n.01')
```

I chose photon and electron because both are very small subatomic particles that exhibit quantum mechanical effects. I think that a score of 0.7 is pretty realistic, given that the particles are still different despite sharing a similar behavior. The lesk algorithm returned the correct synset for photon, given that photon.n.01 describes the photon as the quantum (smallest particle) of the electromagnetic field.

▼ Part 9

Write a couple of sentences about SentiWordNet, describing its functionality and possible use cases. Select an emotionally charged word. Find its senti-synsets and output the polarity scores

for each word. Make up a sentence. Output the polarity for each word in the sentence. Write a couple of sentences about your observations of the scores and the utility of knowing these scores in an NLP application.

SentiWordNet is a library built on top of WordNet that can assign 3 sentiment values to a given synset: positive, negative, and objectivity. These 3 values always add up to one.

```
from nltk.corpus import sentiwordnet as swn

word = 'irate'

senti_sets = swn.senti_synsets(word)
all = list(senti_sets)
print(all)

irate = all[0]

print('Positive score =', irate.pos_score())
print('Negative score =', irate.neg_score())
print('Objective score =', irate.obj_score())

    [SentiSynset('irate.s.01')]
    Positive score = 0.125
    Negative score = 0.5
    Objective score = 0.375

sent = 'I hate those who are irate.'
words = [word for word in word_tokenize(sent) if word.isalpha()]
for word in words:
    all = list(swn.senti_synsets(word))
    print('\n' + word + ' scores:')
    if all:
        sentiment = all[0]
    else:
        print(None)
        continue
    print('Positive score =', sentiment.pos_score())
    print('Negative score =', sentiment.neg_score())
    print('Objective score =', sentiment.obj_score())

    I scores:
    Positive score = 0.0
    Negative score = 0.0
    Objective score = 1.0

    hate scores:
```

```

Positive score = 0.125
Negative score = 0.375
Objective score = 0.5

```

```

those scores:
None

```

```

who scores:
Positive score = 0.0
Negative score = 0.0
Objective score = 1.0

```

```

are scores:
Positive score = 0.0
Negative score = 0.0
Objective score = 1.0

```

```

irate scores:
Positive score = 0.125
Negative score = 0.5
Objective score = 0.375

```

Seeing all of the sentiment scores for these synsets, it seems very easy to naively score words that carry little sentiment, like 'I' or 'are.' However, when it comes to emotion carrying words (actual sentiment), the sentiment scores don't necessarily fit the contexts that they are used in. In this particular case, hate should probably be more negative while irate should be more objective.

▼ Part 10

Write a couple of sentences about what a collocation is. Output collocations for text4, the Inaugural corpus. Select one of the collocations identified by NLTK. Calculate mutual information. Write commentary on the results of the mutual information formula and your interpretation.

A collocation is a combination of words that frequently or conventionally appear together. An example is 'pay attention'; another could be 'dark day.'

```

from nltk.book import *
import math

text = ' '.join(text4.tokens)

my_collocations = text4.collocation_list()
print(my_collocations)

selected = my_collocations[1]
# selected = ['it', 'is']

```

```

string = ' '.join(selected)
num_tokens = len(text4.tokens)

num_joint = text.count(string)
num_x = text.count(selected[0])
num_y = text.count(selected[1])


pxy = num_joint / (num_tokens - 1)
px = num_x / num_tokens
py = num_y / num_tokens

print(pxy, px, py)

pmi = math.log2(pxy / (px * py))
print(pmi)

[('United', 'States'), ('fellow', 'citizens'), ('years', 'ago'), ('four', 'years'), ('Fe
0.001079136690647482 0.04638295367590794 0.033865049934271196
-0.541586104386413

```



The results of the formula for mutual information makes sense. Essentially if a phrase has a higher probability of appearing together than the probability that the words are independent from each other, then we can conclude that the word is a collocation. As such, even words that often appear together (like 'it' and 'is') are not collocations because they appear independently very frequently.

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 5:26 PM

