Python Notes for Linguistics

alvinntnu.github.io/python-notes/corpus/wordnet.html

WordNet

WordNet is a lexical database for the English language, where word senses are connected as a systematic lexical network.

Import

```
from nltk.corpus import wordnet
```

Synsets

A `synset` has several attributes, which can be extracted via its defined methods:

- `synset.name()`
- `synset.definition()`
- `synset.hypernyms()`
- `synset.hyponyms()`
- `synset.hypernym_path()`
- `synset.pos()`

```
syn = wordnet.synsets('walk', pos='v')[0]
print(syn.name())
print(syn.definition())
```

```
walk.v.01
use one's feet to advance; advance by steps
```

```
syn.examples()
```

```
["Walk, don't run!",
 'We walked instead of driving',
 'She walks with a slight limp',
 'The patient cannot walk yet',
 'Walk over to the cabinet']
```

```
syn.hypernyms()
```

```
[Synset('travel.v.01')]
```

```
syn.hypernyms()[0].hyponyms()
```

```
[Synset('accompany.v.02'),
 Synset('advance.v.01'),
 Synset('angle.v.01'),
 Synset('ascend.v.01'),
 Synset('automobile.v.01'),
 Synset('back.v.02'),
 Synset('bang.v.04'),
 Synset('beetle.v.02'),
 Synset('betake_oneself.v.01'),
 Synset('billow.v.02'),
 Synset('bounce.v.03'),
 Synset('breeze.v.02'),
 Synset('caravan.v.01'),
 Synset('career.v.01'),
 Synset('carry.v.36'),
 Synset('circle.v.01'),
 Synset('circle.v.02'),
 Synset('circuit.v.01'),
 Synset('circulate.v.07'),
 Synset('come.v.01'),
 Synset('come.v.11'),
 Synset('crawl.v.01'),
 Synset('cruise.v.02'),
 Synset('derail.v.02'),
 Synset('descend.v.01'),
 Synset('do.v.13'),
 Synset('drag.v.04'),
 Synset('draw.v.12'),
 Synset('drive.v.02'),
 Synset('drive.v.14'),
 Synset('ease.v.01'),
 Synset('fall.v.01'),
 Synset('fall.v.15'),
 Synset('ferry.v.03'),
 Synset('float.v.01'),
 Synset('float.v.02'),
 Synset('float.v.05'),
 Synset('flock.v.01'),
 Synset('fly.v.01'),
 Synset('fly.v.06'),
 Synset('follow.v.01'),
 Synset('follow.v.04'),
 Synset('forge.v.05'),
 Synset('get_around.v.04'),
 Synset('ghost.v.01'),
 Synset('glide.v.01'),
 Synset('go_around.v.02'),
 Synset('hiss.v.02'),
 Synset('hurtle.v.01'),
 Synset('island_hop.v.01'),
 Synset('lance.v.01'),
 Synset('lurch.v.03'),
 Synset('outflank.v.01'),
 Synset('pace.v.02'),
 Synset('pan.v.01'),
 Synset('pass.v.01'),
 Synset('pass_over.v.04'),
 Synset('play.v.09'),
 Synset('plow.v.03'),
 Synset('prance.v.02'),
 Synset('precede.v.04'),
 Synset('precess.v.01'),
 Synset('proceed.v.02'),
 Synset('propagate.v.02'),
 Synset('pursue.v.02'),
 Synset('push.v.09'),
 Synset('raft.v.02'),
 Synset('repair.v.03'),
 Synset('retreat.v.02'),
 Synset('retrograde.v.02'),
 Synset('return.v.01'),
 Synset('ride.v.01'),
 Synset('ride.v.04'),
 Synset('ride.v.10'),
 Synset('rise.v.01'),
 Synset('roll.v.12'),
 Synset('round.v.01'),
 Synset('run.v.11'),
 Synset('run.v.34'),
 Synset('rush.v.01'),
 Synset('scramble.v.01'),
 Synset('seek.v.04'),
 Synset('shuttle.v.01'),
 Synset('sift.v.01'),
 Synset('ski.v.01'),
 Synset('slice_into.v.01'),
 Synset('slither.v.01'),
 Synset('snowshoe.v.01'),
 Synset('speed.v.04'),
 Synset('steamer.v.01'),
 Synset('step.v.01'),
 Synset('step.v.02'),
 Synset('step.v.06'),
 Synset('stray.v.02'),
 Synset('swap.v.02'),
 Synset('swash.v.01'),
 Synset('swim.v.01'),
 Synset('swim.v.05'),
 Synset('swing.v.03'),
 Synset('taxi.v.01'),
 Synset('trail.v.03'),
 Synset('tram.v.01'),
 Synset('transfer.v.06'),
 Synset('travel.v.04'),
 Synset('travel.v.05'),
 Synset('travel.v.06'),
 Synset('travel_by.v.01'),
 Synset('travel_purposefully.v.01'),
 Synset('travel_rapidly.v.01'),
 Synset('trundle.v.01'),
 Synset('turn.v.06'),
 Synset('walk.v.01'),
 Synset('walk.v.10'),
 Synset('weave.v.04'),
 Synset('wend.v.01'),
 Synset('wheel.v.03'),
 Synset('whine.v.01'),
 Synset('whish.v.02'),
 Synset('whisk.v.02'),
 Synset('whistle.v.02'),
 Synset('withdraw.v.01'),
 Synset('zigzag.v.01'),
 Synset('zoom.v.02')]
```

```
syn.hypernym_paths()
```

```
[[Synset('travel.v.01'), Synset('walk.v.01')]]
```

```
syn.pos()
```

```
'v'
```

Lemmas

A `synset` may coreespond to more than one lemma.

```
syn = wordnet.synsets('walk', pos='n')[0]
print(syn.lemmas())
```

```
[Lemma('walk.n.01.walk'), Lemma('walk.n.01.walking')]
```

Check the lemma names.

```
for l in syn.lemmas():
    print(l.name())
```

```
walk
walking
```

Synonyms

```
synonyms = []
for s in wordnet.synsets('run', pos='v'):
    for l in s.lemmas():
        synonyms.append(l.name())
print(len(synonyms))
print(len(set(synonyms)))

print(set(synonyms))
```

```
98
52
{'turn_tail', 'melt_down', 'guide', 'pass', 'be_given', 'run', 'prevail', 'melt',
'track_down', 'escape', 'feed', 'incline', 'hightail_it', 'function',
'head_for_the_hills', 'move', 'break_away', 'lean', 'ladder', 'bunk', 'go', 'hunt',
'play', 'consort', 'range', 'carry', 'ply', 'campaign', 'scarper', 'scat',
'black_market', 'bleed', 'run_away', 'race', 'course', 'lam', 'take_to_the_woods',
'work', 'fly_the_coop', 'tend', 'execute', 'hunt_down', 'persist', 'endure',
'unravel', 'lead', 'run_for', 'draw', 'extend', 'operate', 'flow', 'die_hard'}
```

Antonyms

Some lemmas have antonyms.

The following examples show how to find the antonyms of `good` for its two different senses, `good.n.02` and `good.a.01`.

```
syn1 = wordnet.synset('good.n.02')
syn1.definition()
```

```
'moral excellence or admirableness'
```

```
ant1 = syn1.lemmas()[0].antonyms()[0]
```

```
ant1.synset().definition()
```

```
'the quality of being morally wrong in principle or practice'
```

```
ant1.synset().examples()
```

```
['attempts to explain the origin of evil in the world']
```

```
syn2 = wordnet.synset('good.a.01')
syn2.definition()
```

```
'having desirable or positive qualities especially those suitable for a thing
specified'
```

```
ant2 = syn2.lemmas()[0].antonyms()[0]
```

```
ant2.synset().definition()
```

```
'having undesirable or negative qualities'
```

```
ant2.synset().examples()
```

```
['a bad report card',
 'his sloppy appearance made a bad impression',
 'a bad little boy',
 'clothes in bad shape',
 'a bad cut',
 'bad luck',
 'the news was very bad',
 'the reviews were bad',
 'the pay is bad',
 'it was a bad light for reading',
 'the movie was a bad choice']
```

Wordnet Synset Similarity

With a semantic network, we can also compute the semantic similarty between two synsets based on their distance on the tree.

In particular, this is possible cause all synsets are organized in a hypernym tree.

The recommended distance metric is Wu-Palmer Similarity (i.e., `synset.wup_similarity()`)

```
s1 = wordnet.synset('walk.v.01')
s2 = wordnet.synset('run.v.01')
s3 = wordnet.synset('toddle.v.01')
```

```
s1.wup_similarity(s2)
```

```
0.2857142857142857
```

```
s1.wup_similarity(s3)
```

```
0.8
```

```
s1.common_hypernyms(s3)
```

```
[Synset('travel.v.01'), Synset('walk.v.01')]
```

```
s1.common_hypernyms(s2)
```

```
[Synset('travel.v.01')]
```

Two more metrics for lexical semilarity:

- `synset.path_similarity()`: Path Similarity
- `synset.lch_similarity()`: Leacock Chordorow Similarity