NLP Lab-4
WordNet

Faculty of Computers and Information

Cairo University

NLP-2019

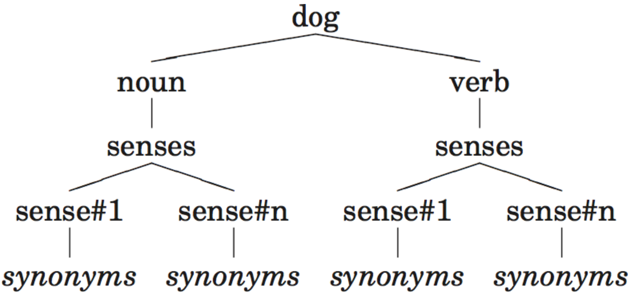
Lab Objectives

It’s a common fact that analyzing text documents is a tough nut to crack for computers. Simple tasks like distinguishing whether a sentence has a positive meaning, or that two words mean literally the same require a have a lot of samples and train various machine learning models.

This lab will show you how you can increase the quality of generated features and gain new insights about your data.

WordNet

- **WordNet** is a database of English words that are linked together by their semantic relationships.
- It is like a supercharged dictionary with a graph structure
- Nouns, verbs, adjectives and adverbs are grouped into sets of synonyms (**synsets**)



Senses and Synonyms

- The first worth-understanding concept is a *“synset”*:
- Synset — “synonym set” — a collection of synonymous words

```
>>> from nltk.corpus import wordnet as wn
>>>wn.synsets("motorcar")
[Synset("car.n.01")]
```

- The output means that word **motorcar** has just one possible context. It is identified by car.n.01 (we will call it "lemma code name") - the first noun (letter n) sense of car.
- You can dig in and see what are other words within this particular synset:

```
>>>wn.synset("car.n.01").lemma_names()
["car","auto","automobile","machine","
        motorcar"]
```

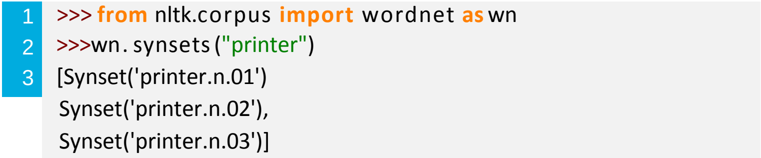
Senses and Synonyms

Synsets are described with a **gloss** (= definition) and some example sentences

```
>>>wn.synset("car.n.01").definition()
"a motor vehicle with four wheels ; usually propelled
by an internal combustion engine"
>>>wn.synset("car.n.01").examples()
["he needs a car to get to work"]
```

Senses and Synonyms

- But as you might expect a word might be ambiguous, for example, a **printer**:



```
>>> from nltk.corpus import wordnet as wn
>>>wn.synsets("printer")
[Synset('printer.n.01')
 Synset('printer.n.02'),
 Synset('printer.n.03')]
```

You see that there are 3 possible contexts. To help understand the meaning of each one we can see it’s definition and provided examples (if are available).

```
>>> for synset in wn.synsets('printer'):
...       print("\tLemma: {}".format(synset.name()))
...       print("\tDefinition: {}".format(synset.definition()))

  Lemma: printer.n.01
  Definition: someone whose occupation is printing

  Lemma: printer.n.02
  Definition: (computer science) an output device that prints the results of data processing

  Lemma: printer.n.03
  Definition: a machine that prints
```

Senses and Synonyms

Unlike the words automobile and motorcar, which are unambiguous and have one synset, the word **car** is ambiguous, having five synsets:

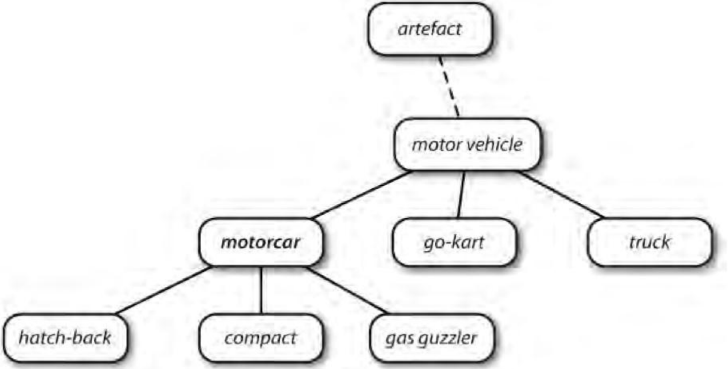
```
>>>wn.synsets("car")
[ Synset ("car.n.01") , Synset("car.n.02") , Synset("car.n.03") , Synset ("car.n.04") , Synset ("cable_car.n.01") ]
>>> for synset in wn.synsets("car") :
...    print synset .lemma_names()
...
["car","auto","automobile","machine",“motorcar”]
["car","railcar","railway_car","railroad_car"]
["car","gondola"]
["car","elevator_car"]
["cable_car","car"]
```

The WordNet Hierarchy

Lexical relations

Hypernyms and hyponyms ("is-a relation")

- *motor vehicle* is a hypernym of *motorcar*
- *ambulance* is a hyponym of *motorcar*



The WordNet Hierarchy - Child

```
>>> motorcar = wn.synset("car.n.01")
>>> types_of_motorcar = motorcar.hyponyms()
>>> sorted([lemma.name() for synset in types_of_motorcar
    for lemma in synset.lemmas()])

 ["Model_T","S.U.V.","SUV","Stanley_Steamer","ambulance"
        ,"beach_waggon","beach_wagon","bus","cab","
        compact","compact_car","convertible","coupe"," cruis
        er","electric","electric_automobile","
        electric_car","estate_car","gas_guzzler","hack"," hardto
        p","hatchback","heap","horseless_carriage"," hot−rod","hot_
        rod","jalopy","jeep","landrover","limo","limousine","loane
        r","minicar","minivan",
        " pace_car","patrol_car","phaeton","police_car","
        police_cruiser","prowl_car","race_car","racer","
        racing_car"...]
```

The WordNet Hierarchy - parent

```
>>> motorcar = wn.synset("car.n.01")
```

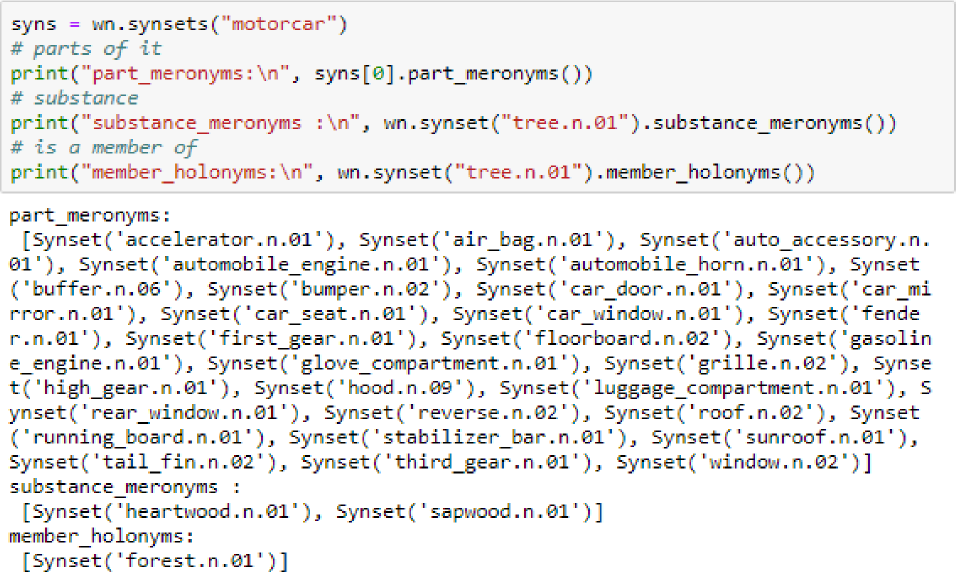
```
>>> motorcar . hypernyms ( )
      [ Synset ("motor_vehicle.n.01") ]
>>> paths = motorcar . hypernym_paths ( )
>>> len ( paths )
       2
>>> [ synset . name ( )       for synset in paths [ 0 ] ]
 ['entity.n.01', 'physical_entity.n.01', 'object.n.01', 'whole.n.02',
'artifact.n.01', 'instrumentality.n.03', 'container.n.01',
'wheeled_vehicle.n.01', 'self-propelled_vehicle.n.01', 'motor_vehicle.n.01',
'car.n.01']


>>> [ synset . name ( )   for synset in paths [ 1 ] ]
['entity.n.01', 'physical_entity.n.01', 'object.n.01', 'whole.n.02',
'artifact.n.01', 'instrumentality.n.03', 'conveyance.n.03', 'vehicle.n.01',
'wheeled_vehicle.n.01', 'self-propelled_vehicle.n.01', 'motor_vehicle.n.01',
'car.n.01']
```

More Lexical Relations

Meronyms and holonyms

- *branch* is a meronym (*part meronym*) of *tree.*
- *heartwood* is a meronym (*substance meronym*) of *tree.*
- *forest* is a holonym (*member holonym*) of *tree.*

More Lexical Relations



```
syns = wn.synsets("motorcar")
# parts of it
print("part_meronyms:\n", syns[0].part_meronyms())
# substance
print("substance_meronyms :\n", wn.synset("tree.n.01").substance_meronyms())
# is a member of
print("member_holonyms:\n", wn.synset("tree.n.01").member_holonyms())
```

```
part_meronyms:
 [Synset('accelerator.n.01'), Synset('air_bag.n.01'), Synset('auto_accessory.n.
01'), Synset('automobile_engine.n.01'), Synset('automobile_horn.n.01'), Synset
('buffer.n.06'), Synset('bumper.n.02'), Synset('car_door.n.01'), Synset('car_mi
rror.n.01'), Synset('car_seat.n.01'), Synset('car_window.n.01'), Synset('fende
r.n.01'), Synset('first_gear.n.01'), Synset('floorboard.n.02'), Synset('gasolin
e_engine.n.01'), Synset('glove_compartment.n.01'), Synset('grille.n.02'), Synse
t('high_gear.n.01'), Synset('hood.n.09'), Synset('luggage_compartment.n.01'), S
ynset('rear_window.n.01'), Synset('reverse.n.02'), Synset('roof.n.02'), Synset
('running_board.n.01'), Synset('stabilizer_bar.n.01'), Synset('sunroof.n.01'),
Synset('tail_fin.n.02'), Synset('third_gear.n.01'), Synset('window.n.02')]
substance_meronyms :
 [Synset('heartwood.n.01'), Synset('sapwood.n.01')]
member_holonyms:
 [Synset('forest.n.01')]
```

More Lexical Relations

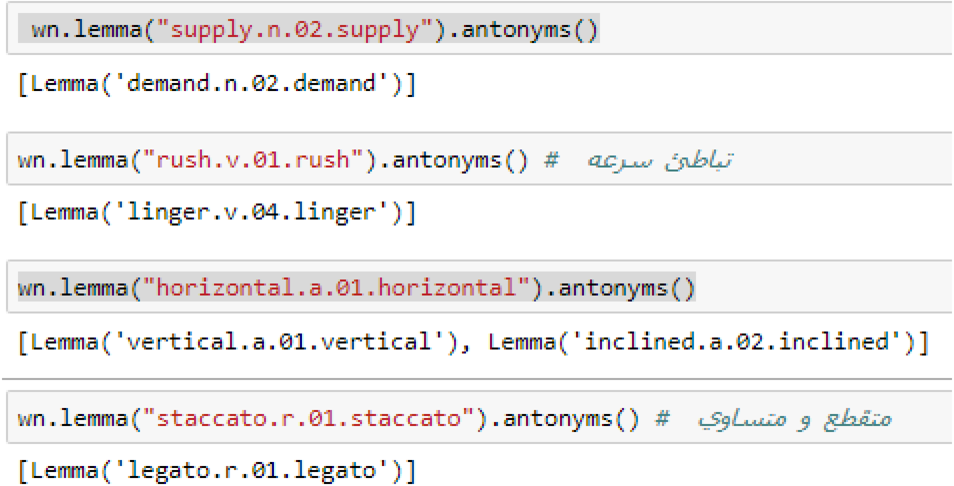
Relationships between verbs:

- the act of walking involves the act of stepping, so walking entails stepping
- some verbs have multiple entailments

```
>>>wn.synset("walk.v.01").entailments()
[Synset("step.v.01")]
>>>wn.synset("eat.v.01").entailments()
[Synset ("swallow.v.01") , Synset ("chew.v.01")]
```

More Lexical Relations - antonyms - المتضادات

Some lexical relationships can express antonymy:



```
wn.lemma("supply.n.02.supply").antonyms()
```

```
[Lemma('demand.n.02.demand')]
```

```
wn.lemma("rush.v.01.rush").antonyms() #  تباطئ سرعه
```

```
[Lemma('linger.v.04.linger')]
```

```
wn.lemma("horizontal.a.01.horizontal").antonyms()
```

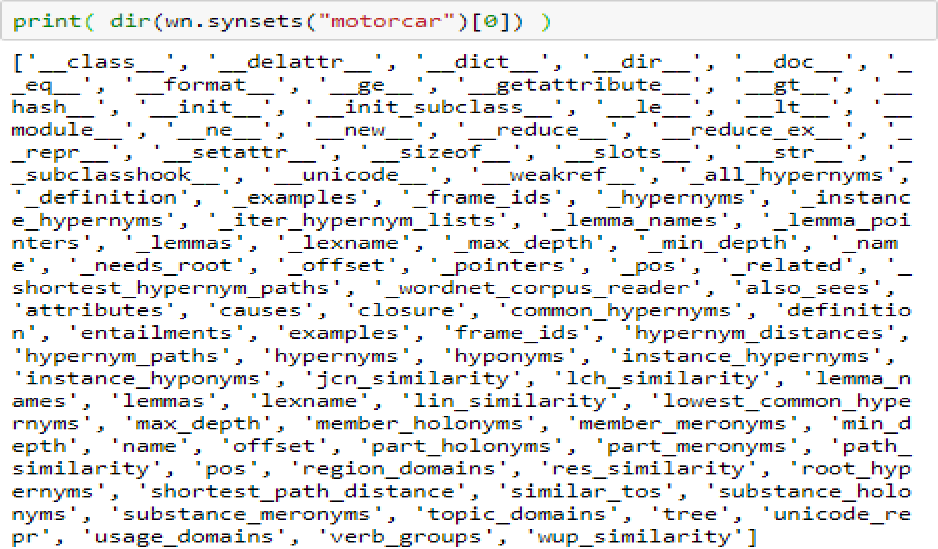
```
[Lemma('vertical.a.01.vertical'), Lemma('inclined.a.02.inclined')]
```

```
wn.lemma("staccato.r.01.staccato").antonyms() #  متقطع و متساوي
```

```
[Lemma('legato.r.01.legato')]
```

More Lexical Relations

You can see the lexical relations, and the other methods defined on a synset, using `dir()`. For example:



```
print( dir(wn.synsets("motorcar")[0]) )
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__
hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__
module__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '_
_repr__', '__setattr__', '__sizeof__', '__slots__', '__str__', '_
_subclasshook__', '__unicode__', '__weakref__', '_all_hypernyms',
'_definition', '_examples', '_frame_ids', '_hypernyms', '_instanc
e_hypernyms', '_iter_hypernym_lists', '_lemma_names', '_lemma_poi
nters', '_lemmas', '_lexname', '_max_depth', '_min_depth', '_nam
e', '_needs_root', '_offset', '_pointers', '_pos', '_related', '_
shortest_hypernym_paths', '_wordnet_corpus_reader', 'also_sees',
'attributes', 'causes', 'closure', 'common_hypernyms', 'definitio
n', 'entailments', 'examples', 'frame_ids', 'hypernym_distances',
'hypernym_paths', 'hypernyms', 'hyponyms', 'instance_hypernyms',
'instance_hyponyms', 'jcn_similarity', 'lch_similarity', 'lemma_n
ames', 'lemmas', 'lexname', 'lin_similarity', 'lowest_common_hype
rnyms', 'max_depth', 'member_holonyms', 'member_meronyms', 'min_d
epth', 'name', 'offset', 'part_holonyms', 'part_meronyms', 'path_
similarity', 'pos', 'region_domains', 'res_similarity', 'root_hyp
ernyms', 'shortest_path_distance', 'similar_tos', 'substance_holo
nyms', 'substance_meronyms', 'topic_domains', 'tree', 'unicode_re
pr', 'usage_domains', 'verb_groups', 'wup_similarity']
```

Semantic Similarity

Two synsets linked to the same root may have several hypernyms in common. If two synsets share a very specific hypernym (low down in the hypernym hierarchy), they must be closely related.

```
>>> truck = wn.synset('truck.n.01')
>>> limousine = wn.synset('limousine.n.01')
>>> truck.lowest_common_hypernyms(limousine)
[Synset('motor_vehicle.n.01')]

>>> right_whale= wn.synset('right_whale.n.01')
>>> tortoise = wn.synset('tortoise.n.01')
>>> right_whale.lowest_common_hypernyms(tortoise)
[Synset('vertebrate.n.01')]   // حيوان فقاري

>>>wn.synset('policeman.n.01').lowest_common_hypernyms(
wn.synset('chef.n.01'))
[Synset('person.n.01')]
```

Semantic Similarity

We can quantify this concept of generality by looking up the depth of each synset:

```
>>>wn.synset("baleen_whale.n.01").min_depth()
14
>>>wn.synset("whale.n.02").min_depth()
13
>>>wn.synset("vertebrate.n.01").min_depth()
8
>>>wn.synset("entity.n.01").min_depth()
0
```

Semantic similarity

Similarity measures have been defined over the collection of WordNet synsets that incorporate this insight

- path_similarity() assigns a score in the range 0-1 based on the shortest path that connects the concepts in the hypernym hierarchy
- -1 is returned in those cases where a path cannot be found
- Comparing a synset with itself will return 1

Semantic Similarity

```
>>>tortoise.path_similarity(right_whale)
0.07692307692307693
>>>tortoise.path_similarity(tortoise)
1.0
>>>truck.path_similarity(limousine)
0.25
>>>truck.path_similarity(truck)
1.0
```

Polysemy

The **polysemy** of a word is the number of senses it has. The noun **dog** has 7 senses in WordNet:

```python
from nltk.corpus import wordnet as wn
num_senses=len(wn.synsets("dog","n"))

print(num_senses)
#prints 7
```

We can also compute the average polysemy of nouns, verbs, adjectives and adverbs according to WordNet.

More on Wordnet

More about WordNet

```
from nltk.corpus import wordnet as wn
>>> wn.synset('code.n.03').topic_domains()
 [Synset('computer_science.n.01')]
 >>> wn.synset('pukka.a.01').region_domains() # أصلي
[Synset('india.n.01')]  # = to original in english
>>> wn.synset('freaky.a.01').usage_domains()   # فظيع
[Synset('slang.n.02')] # عامية
```

More on Wordnet

More about WordNet

```
from nltk.corpus import wordnet as wn

>>> wn.synsets('book', wn.NOUN)

[Synset('book.n.01'), Synset('book.n.02'), Synset('record.n.05'), Synset('script.n.01'),
Synset('ledger.n.01'), Synset('book.n.06'), Synset('book.n.07'), Synset('koran.n.01'),
Synset('bible.n.01'), Synset('book.n.10'), Synset('book.n.11')]

>>> wn.synsets('book', wn.ADJ)

[]
```

References

- https://sp1718.github.io/wordnet_lecture.pdf
- https://medium.com/parrot-prediction/dive-into-wordnet-with-nltk-b313c480e788
- http://www.nltk.org/howto/wordnet.html