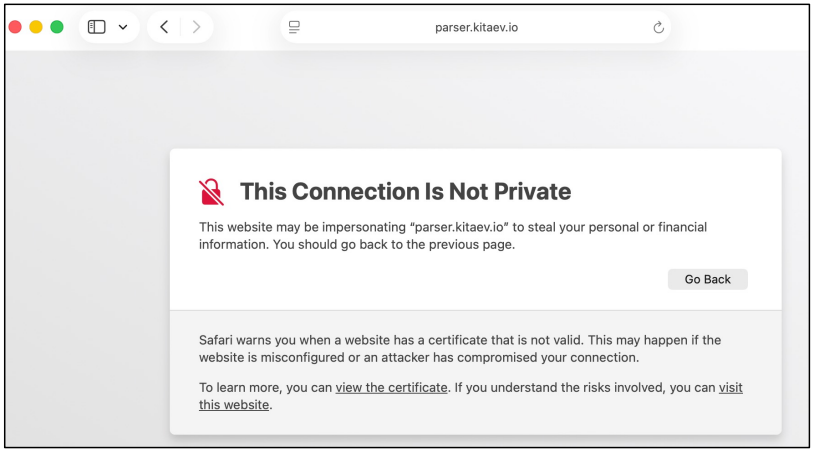
LING/C SC 581:
Advanced Computational Linguistics

Lecture 16

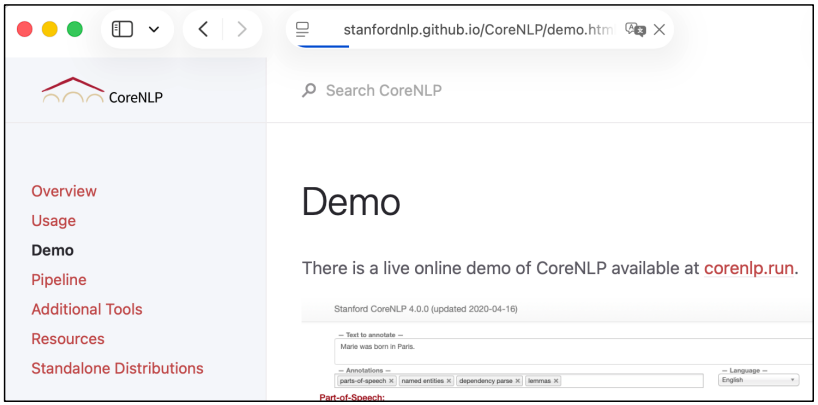
Prof. Sandiway Fong

Online Parser Access

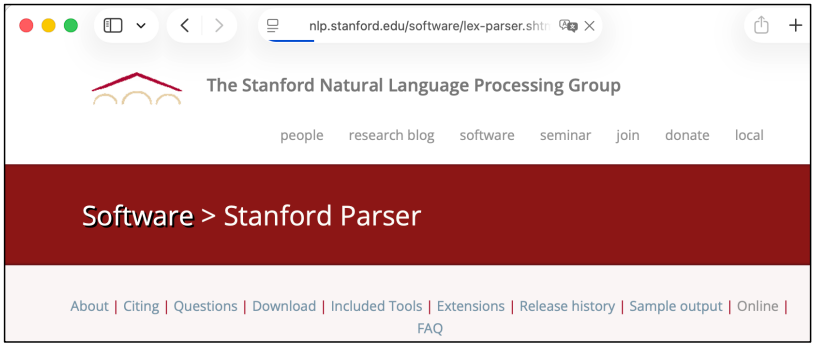
Berkeley Neural Parser



CoreNLP



Stanford Parser



Today's Topic

- WordNet: introduction using nltk
  - synsets
  - relations
  - morphy
- Later: a look at similarity
  - WordNet
  - Word Embeddings

nltk: WordNet

References

- Browser: http://wordnetweb.princeton.edu/perl/webwn
- nltk: http://www.nltk.org/howto/wordnet.html
- https://www.nltk.org/api/nltk.corpus.reader.wordnet.html

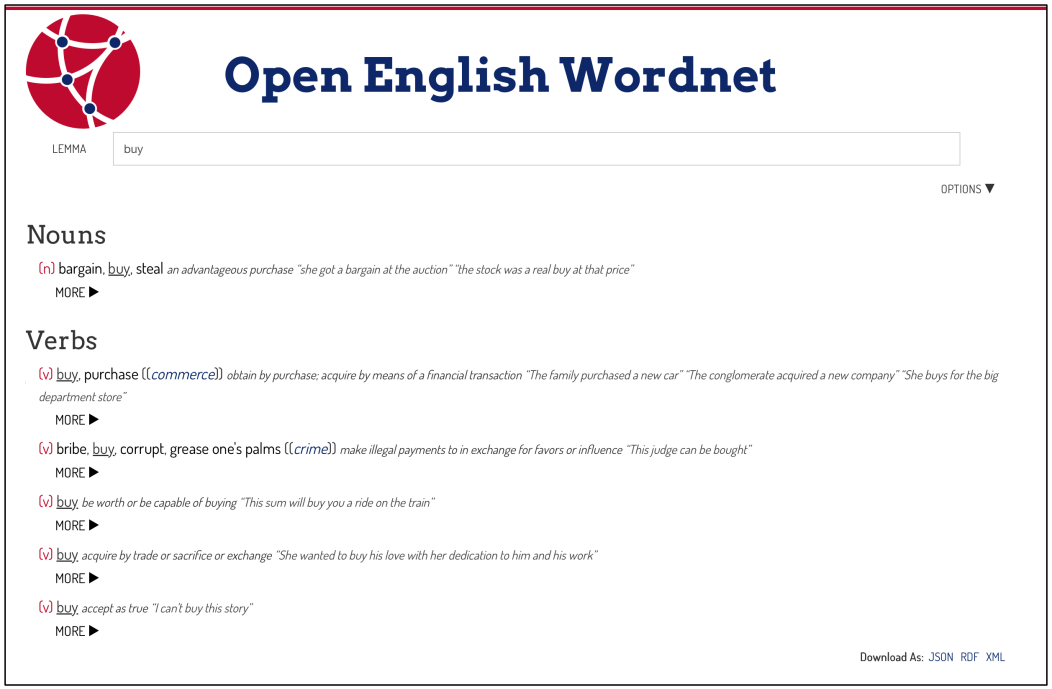
No longer supported

Python

```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('table')
[Synset('table.n.01'), Synset('table.n.02'), Synset('table.n.03'),
Synset('mesa.n.01'), Synset('table.n.05'), Synset('board.n.04'),
Synset('postpone.v.01'), Synset('table.v.02')]
>>> wn.synset('table.n.02')
Synset('table.n.02')
```

https://en-word.net

https://github.com/globalwordnet/english-wordnet/blob/main/README.md



Princeton WordNet

WordNet® is a large lexical database of English.

- Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (**synsets**), each expressing a distinct concept.
- **Synsets** are interlinked by means of conceptual-semantic and lexical **relations**.
- The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download.
- WordNet's structure makes it a useful tool for computational linguistics and natural language processing.

Princeton WordNet

- WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings.
- However, there are some important distinctions.
  - First, WordNet interlinks not just word forms—strings of letters—but **specific senses** of words.
  - As a result, words that are found in close proximity to one another in the network are semantically disambiguated.
  - Second, WordNet labels the **semantic relations** among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity.

Princeton WordNet

- The main relation among words in WordNet is *synonymy* (**synset**):
  - *shut* and *close*
  - *car* and *automobile*
- nltk:

Synset: <word>.<pos>.<number>
Lemma: <word>.<pos>.<number>.<lemma>

```
>>> wn.synsets('shut')
[Synset('close.v.01'), Synset('close.v.02'), Synset('exclude.v.02'),
  Synset('shut.a.01'), Synset('closed.a.04')]
>>> wn.synsets('shut')[0].lemmas()
[Lemma('close.v.01.close'), Lemma('close.v.01.shut')]
>>> wn.synsets('shut')[1].lemmas()
[Lemma('close.v.02.close'), Lemma('close.v.02.shut')]
>>> wn.synsets('shut')[0].lemma_names()
['close', 'shut']
```

Princeton WordNet

synsets(*lemma, pos=None, lang='eng', check_exceptions=True*) **[source]**

Load all synsets with a given lemma and part of speech tag. If no pos is specified, all synsets for all parts of speech will be loaded. If lang is specified, all the synsets associated with the lemma name of that language will be returned.

synset(*name*)

pos (*str*) – The Part Of Speech tag.
- *"n"* for nouns,
- *"v"* for verbs,
- *"a"* for adjectives,
- *"r"* for adverbs and
- *"s"* for satellite adjectives.

```
>>> wn.synset('shut.v.01')
Synset('close.v.01')
>>> wn.synset('shut.v')
ValueError: not enough values to unpack (expected 3, got 2)
>>> wn.synset('shut')
ValueError: not enough values to unpack (expected 3, got 1)
```

Princeton WordNet

definition(*lang='eng'*) [source]

Return definition in specified language

examples(*lang='eng'*) [source]

Return examples in specified language

```
>>> wn.synset('close.v.01').definition()
'move so that an opening or passage is obstructed; make shut'
>>> wn.synset('close.v.02').definition()
'become closed'
>>> wn.synset('close.v.01').examples()
['Close the door', 'shut the window']
>>> wn.synset('close.v.02').examples()
['The windows closed with a loud bang']
```

transitive

intransitive

Also ok with:
`'shut.v.01'`
`'shut.v.02'`

Princeton WordNet

- The main relation among words in WordNet is *synonymy* (**synset**):
  - *shut* and *close*
- Overlap:

```
>>> wn.synsets('shut')
```

2 out of 5 senses

```
[Synset('close.v.01'), Synset('close.v.02'), Synset('exclude.v.02'), Synset('shut.a.01'),
  Synset('closed.a.04')]
```

```
>>> wn.synsets('close')
```

2 out of 37 senses

```
[Synset('stopping_point.n.01'), Synset('conclusion.n.08'), Synset('finale.n.03'),
  Synset('close.v.01'), Synset('close.v.02'), Synset('close_up.v.01'),
  Synset('close.v.04'), Synset('conclude.v.04'), Synset('close.v.06'),
  Synset('close.v.07'), Synset('close.v.08'), Synset('close.v.09'), Synset('close.v.10'),
  Synset('close.v.11'), Synset('close.v.12'), Synset('close.v.13'), Synset('close.v.14'),
  Synset('close.v.15'), Synset('close_up.v.03'), Synset('close.v.17'),
  Synset('close.a.01'), Synset('close.a.02'), Synset('near.a.01'), Synset('close.s.04'),
  Synset('close.s.05'), Synset('close.s.06'), Synset('close.s.07'),
  Synset('airless.s.01'), Synset('close.s.09'), Synset('close.s.10'),
  Synset('close.s.11'), Synset('close.s.12'), Synset('close.s.13'),
  Synset('cheeseparing.s.01'), Synset('close.s.15'), Synset('near.r.01'),
  Synset('close.r.02')]
```

Merriam Webster

Q: *How many senses*?

shut 1 of 3 verb

shət

shut; shutting

Synonyms of *shut* ›

transitive verb

1 a : to move into position to close an opening
shut the lid

b : to prevent entrance to or passage to or from

2 : to confine by or as if by enclosure
shut herself in her study

3 : to fasten with a lock or bolt

4 : to close by bringing enclosing or covering parts together
shut the eyes

5 : to cause to cease or suspend an operation or activity → often used with *down*

intransitive verb

1 : to close itself or become closed
flowers that *shut* at night

2 : to cease or suspend an operation or activity → often used with *down*

shut 2 of 3 adjective

1 : closed, fastened, or folded together

2 : RID, CLEAR, FREE → usually used with *of*

shut 3 of 3 noun

: the act of shutting

```
wn.synset('close.v.01').definition()
'move so that an opening or passage is obstructed; make shut'
```

```
wn.synset('close.v.02').definition()
'become closed'
```

close 1 of 5 verb

transitive verb

1 a : to move so as to bar passage through something
Close the gate.

b : to block against entry or passage
close a street

c : to deny access to
The city *closed* the beach.

d : to suspend or stop the operations of
close school
→ often used with *down*
They *closed* down the factory.

e : SCREEN, EXCLUDE

closed his football career with an outstanding big bowl performance
Investigators *closed* the case after concluding that his death was accidental.

b : to terminate access to (a computer file or program)
Remember to save the file before *closing* it.

c : to conclude discussion or negotiation about
The question is *closed*.

also **:** to consummate (see CONSUMMATE entry 2 sense 2) by performing something previously agreed
close a transfer of real estate title

3 a : to bring or bind together the parts or edges of
a *closed* book

b : to reduce to nil
closed the distance to the lead racer

c : to fill up (something, such as an opening)
close the cracks with plaster of paris

d : to make complete by circling or enveloping or by making continuous
close a circuit

4 archaic **:** ENCLOSE, CONTAIN

or slide so as to leave no opening

b : to cease operation
The factory *closed* down.
The stores *close* at 9 p.m.

2 a : to come together **:** MEET
The jaws of the vise *closed*.

b : to draw the free foot up to the supporting foot in dancing

3 : to come to an end or period
The services *closed* with a short prayer.

4 : to enter into or complete an agreement
close on a deal

5 : to reduce a gap
closed to within two points

6 a : to draw near
The ship was *closing* with the island.

b : to engage in a struggle at close quarters **:** GRAPPLE
close with the enemy

closable adjective

variants *or* **closeable** klō-zə-bəl

Princeton WordNet

- *The most frequently encoded **relation** among **synsets** is the super-subordinate relation* (*also called **hyperonymy**, **hyponymy** or **ISA** relation*).
  - links {*furniture*, *piece_of_furniture*} to specific ones like {*bed*} and {*bunkbed*}.
  - WordNet distinguishes among **Types** (common nouns) and **Instances** (specific persons, countries and geographic entities).
  - Thus, *armchair* is a **type** of *chair*, *Barack Obama* is an **instance** of a *president*.
  - **Instances** are always leaf (**terminal**) nodes in their hierarchies.

Princeton WordNet

- Synsets have the following methods for retrieving related Synsets.
  - hypernyms, instance_hypernyms
  - hyponyms, instance_hyponyms

```
>>> wn.synsets('entity')
[Synset('entity.n.01')]
>>> wn.synset('entity.n.01').hyponyms()
[Synset('abstraction.n.06'), Synset('physical_entity.n.01'), Synset('thing.n.08')]
>>> wn.synset('entity.n.01').hypernyms()
[]
>>> wn.synset('entity.n.01').instance_hyponyms()
[]
>>> wn.synsets('Tucson')
[Synset('tucson.n.01')]
>>> wn.synset('tucson.n.01').instance_hypernyms()
[Synset('city.n.01')]
>>> wn.synset('tucson.n.01').hypernyms()
[]
```

Princeton WordNet

closure(*rel*, *depth=-1*) [source]

Return the transitive closure of source under the rel relationship, breadth-first, discarding cycles:

```
>>> from nltk.corpus import wordnet as wn
```

```
>>> list(wn.synset('entity.n.01').closure(lambda x:x.hyponyms(),depth=2))
[Synset('abstraction.n.06'), Synset('physical_entity.n.01'),
Synset('thing.n.08'), Synset('relation.n.01'), Synset('measure.n.02'),
Synset('communication.n.02'), Synset('set.n.02'), Synset('attribute.n.02'),
Synset('psychological_feature.n.01'), Synset('otherworld.n.01'),
Synset('group.n.01'), Synset('thing.n.12'), Synset('process.n.06'),
Synset('causal_agent.n.01'), Synset('substance.n.04'), Synset('matter.n.03'),
Synset('object.n.01'), Synset('change.n.06'), Synset('stinker.n.02'),
Synset('pacifier.n.02'), Synset('whacker.n.01'), Synset('security_blanket.n.01'),
Synset('freshener.n.01'), Synset('jimdandy.n.02'), Synset('horror.n.02')]
```

Princeton WordNet

tree(*rel*, *depth=-1*, *cut_mark=None*) [source]

Return the full relation tree, including self, discarding cycles:

```
>>> list(wn.synset('entity.n.01').tree(lambda x:x.hyponyms(),depth=2))
[Synset('entity.n.01'), [Synset('abstraction.n.06'),
[Synset('relation.n.01')], [Synset('measure.n.02')],
[Synset('communication.n.02')], [Synset('set.n.02')],
[Synset('attribute.n.02')], [Synset('psychological_feature.n.01')],
[Synset('otherworld.n.01')], [Synset('group.n.01')]],
[Synset('physical_entity.n.01'), [Synset('thing.n.12')],
[Synset('process.n.06')], [Synset('causal_agent.n.01')],
[Synset('substance.n.04')], [Synset('matter.n.03')],
[Synset('object.n.01')]], [Synset('thing.n.08'), [Synset('change.n.06')],
[Synset('stinker.n.02')], [Synset('pacifier.n.02')],
[Synset('whacker.n.01')], [Synset('security_blanket.n.01')],
[Synset('freshener.n.01')], [Synset('jimdandy.n.02')],
[Synset('horror.n.02')]]]
```

Princeton WordNet

lowest_common_hypernyms(*other*, *simulate_root=False*, *use_min_depth=False*) **[source]**

Get a list of lowest synset(s) that both synsets have as a hypernym. When use_min_depth == False this means that the synset which appears as a hypernym of both self and other with the lowest maximum depth is returned or if there are multiple such synsets at the same depth they are all returned

```
>>> wn.synsets('furniture')
[Synset('furniture.n.01')]
>>> wn.synsets('bed','n')
[Synset('bed.n.01'), Synset('bed.n.02'), Synset('bed.n.03'), Synset('bed.n.04'),
Synset('seam.n.03'), Synset('layer.n.01'), Synset('bed.n.07'),
Synset('bed.n.08')]
>>> wn.synsets('furniture')[0].lowest_common_hypernyms(wn.synset('bed.n.01'))
[Synset('furniture.n.01')]
>>> wn.synsets('furniture')[0].lowest_common_hypernyms(wn.synset('bed.n.02'))
[Synset('object.n.01')]
```

Princeton WordNet

digraph(*inputs, rel=<function WordNetCorpusReader.<lambda>>, pos=None, maxdepth=-1,*

```
>>> print(wn.digraph([wn.synset('entity.n.01')],rel=lambda x:x.hyponyms(),maxdepth=2))
digraph G {
"Synset('abstraction.n.06')" -> "Synset('attribute.n.02')";
"Synset('abstraction.n.06')" -> "Synset('communication.n.02')";
"Synset('abstraction.n.06')" -> "Synset('group.n.01')";
"Synset('abstraction.n.06')" -> "Synset('measure.n.02')";
"Synset('abstraction.n.06')" -> "Synset('otherworld.n.01')";
"Synset('abstraction.n.06')" -> "Synset('psychological_feature.n.01')";
"Synset('abstraction.n.06')" -> "Synset('relation.n.01')";
"Synset('abstraction.n.06')" -> "Synset('set.n.02')";
"Synset('entity.n.01')" -> "Synset('abstraction.n.06')";
"Synset('entity.n.01')" -> "Synset('physical_entity.n.01')";
"Synset('entity.n.01')" -> "Synset('thing.n.08')";
"Synset('physical_entity.n.01')" -> "Synset('causal_agent.n.01')";
"Synset('physical_entity.n.01')" -> "Synset('matter.n.03')";
"Synset('physical_entity.n.01')" -> "Synset('object.n.01')";
"Synset('physical_entity.n.01')" -> "Synset('process.n.06')";
"Synset('physical_entity.n.01')" -> "Synset('substance.n.04')";
"Synset('physical_entity.n.01')" -> "Synset('thing.n.12')";
"Synset('thing.n.08')" -> "Synset('change.n.06')";
"Synset('thing.n.08')" -> "Synset('freshener.n.01')";
"Synset('thing.n.08')" -> "Synset('horror.n.02')";
"Synset('thing.n.08')" -> "Synset('jimdandy.n.02')";
"Synset('thing.n.08')" -> "Synset('pacifier.n.02')";
"Synset('thing.n.08')" -> "Synset('security_blanket.n.01')";
"Synset('thing.n.08')" -> "Synset('stinker.n.02')";
"Synset('thing.n.08')" -> "Synset('whacker.n.01')";}
```

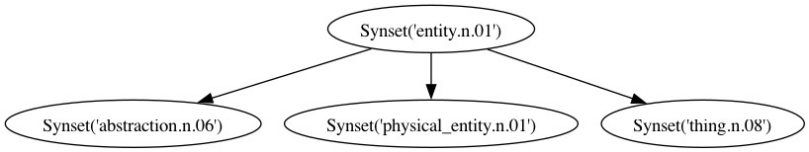
Princeton WordNet

```
dot -Tpng digraph.dot > digraph.png
```

```
digraph G {
rankdir="LR";
"Synset('abstraction.n.06')" -> "Synset('attribute.n.02')";
"Synset('abstraction.n.06')" -> "Synset('communication.n.02')";
"Synset('abstraction.n.06')" -> "Synset('group.n.01')";
"Synset('abstraction.n.06')" -> "Synset('measure.n.02')";
"Synset('abstraction.n.06')" -> "Synset('otherworld.n.01')";
"Synset('abstraction.n.06')" ->
"Synset('psychological_feature.n.01')";
"Synset('abstraction.n.06')" -> "Synset('relation.n.01')";
"Synset('abstraction.n.06')" -> "Synset('set.n.02')";
"Synset('entity.n.01')" -> "Synset('abstraction.n.06')";
"Synset('entity.n.01')" -> "Synset('physical_entity.n.01')";
"Synset('entity.n.01')" -> "Synset('thing.n.08')";
"Synset('physical_entity.n.01')" -> "Synset('causal_agent.n.01')";
"Synset('physical_entity.n.01')" -> "Synset('matter.n.03')";
"Synset('physical_entity.n.01')" -> "Synset('object.n.01')";
"Synset('physical_entity.n.01')" -> "Synset('process.n.06')";
"Synset('physical_entity.n.01')" -> "Synset('substance.n.04')";
"Synset('physical_entity.n.01')" -> "Synset('thing.n.12')";
"Synset('thing.n.08')" -> "Synset('change.n.06')";
"Synset('thing.n.08')" -> "Synset('freshener.n.01')";
"Synset('thing.n.08')" -> "Synset('horror.n.02')";
"Synset('thing.n.08')" -> "Synset('jimdandy.n.02')";
"Synset('thing.n.08')" -> "Synset('pacifier.n.02')";
"Synset('thing.n.08')" -> "Synset('security_blanket.n.01')";
"Synset('thing.n.08')" -> "Synset('stinker.n.02')";
"Synset('thing.n.08')" -> "Synset('whacker.n.01')";
}
```

Princeton WordNet

- All noun hierarchies ultimately go up the root node {entity}.



- **Exercise 1**: find all the lowest common hypernyms between senses of *furniture* and *bed*
- **Exercise 2**: find all the lowest common hypernyms between senses of *furniture* and *bunkbed*

Whiteboard