LING/C SC 581:

Advanced Computational Linguistics

Lecture 17

nltk: WordNet

References

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

python

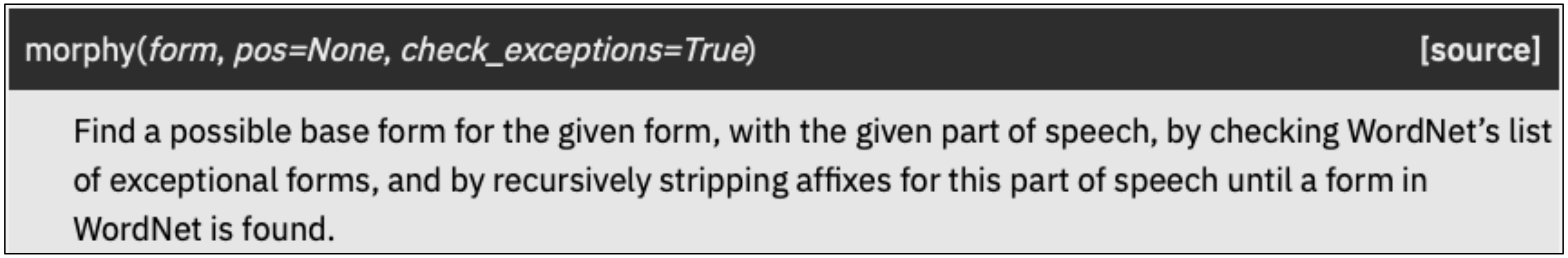
```
>>> from nltk.corpus import wordnet as wn
>>> wn.synsets('table')
[Synset('table.n.01'), Synset('table.n.02'), Synset('table.n.03'),
Synset('mesa.n.01'), Synset('table.n.05'), Synset('board.n.04'),
Synset('postpone.v.01'), Synset('table.v.02')]
>>> wn.synset('table.n.02')
Synset('table.n.02')
```

Today's Topic

- WordNet
  - `morphy`: inflected form to base form
  - used in browser

- A look at similarity
  - WordNet
  - Word Embeddings

wn.mophy()



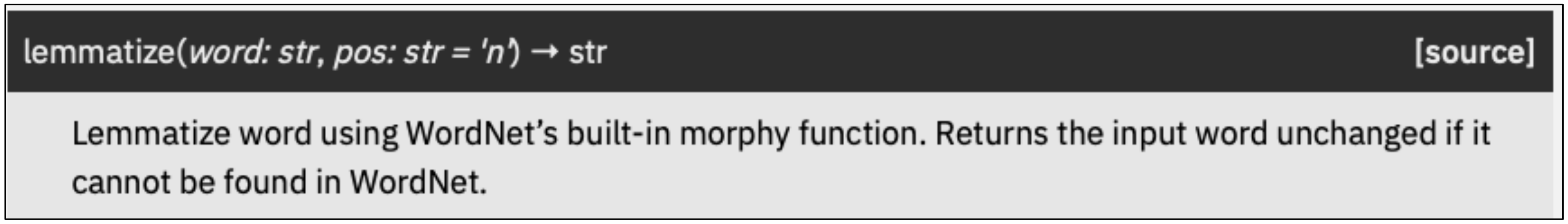
morphy(*form, pos=None, check_exceptions=True*) [source]

Find a possible base form for the given form, with the given part of speech, by checking WordNet's list of exceptional forms, and by recursively stripping affixes for this part of speech until a form in WordNet is found.

```
>>> print(wn.morphy('dogs')) dog
>>> print(wn.morphy('churches')) church
>>> print(wn.morphy('aardwolves')) aardwolf
>>> print(wn.morphy('abaci')) abacus
>>> wn.morphy('hardrock', wn.ADV)
>>> print(wn.morphy('book', wn.NOUN)) book
 >>> wn.morphy('book', wn.ADJ)
```

WordNetLemmatizer

https://www.nltk.org/api/nltk.stem.wordnet.html



lemmatize(*word: str, pos: str = 'n'*) → str **[source]**

Lemmatize word using WordNet's built-in morphy function. Returns the input word unchanged if it cannot be found in WordNet.

```
>>> from nltk.stem import WordNetLemmatizer
>>> wnl = WordNetLemmatizer()
>>> print(wnl.lemmatize('dogs')) dog
>>> print(wnl.lemmatize('churches')) church
>>> print(wnl.lemmatize('aardwolves')) aardwolf
>>> print(wnl.lemmatize('abaci')) abacus
>>> print(wnl.lemmatize('hardrock')) hardrock
```

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

WordNet: *table*

The noun table has 6 senses (first 3 from tagged texts)

1. (52) table#1, tabular array#1 -- (a set of data arranged in rows and columns; "see table 1")

2. (25) table#2 -- (a piece of furniture having a smooth flat top that is usually supported by one or more vertical legs; "it was a sturdy table")

3. (5) table#3 -- (a piece of furniture with tableware for a meal laid out on it; "I reserved a table at my favorite restaurant")

4. mesa#1, table#4 -- (flat tableland with steep edges; "the tribe was relatively safe on the mesa but they had to descend into the valley for water")

5. table#5 -- (a company of people assembled at a table for a meal or game; "he entertained the whole table with his witty remarks")

6. board#4, table#6 -- (food or meals in general; "she sets a fine table"; "room and board")

The verb table has 2 senses (no senses from tagged texts)

1. postpone#1, prorogue#1, hold over#5, put over#2, table#1, shelve#1, set back#1, defer#1, remit#2, put off#1 -- (hold back to a later time; "let's postpone the exam")

2. table#2, tabularize#1, tabularise#1, tabulate#1 -- (arrange or enter in tabular form)

WordNet: *stool*

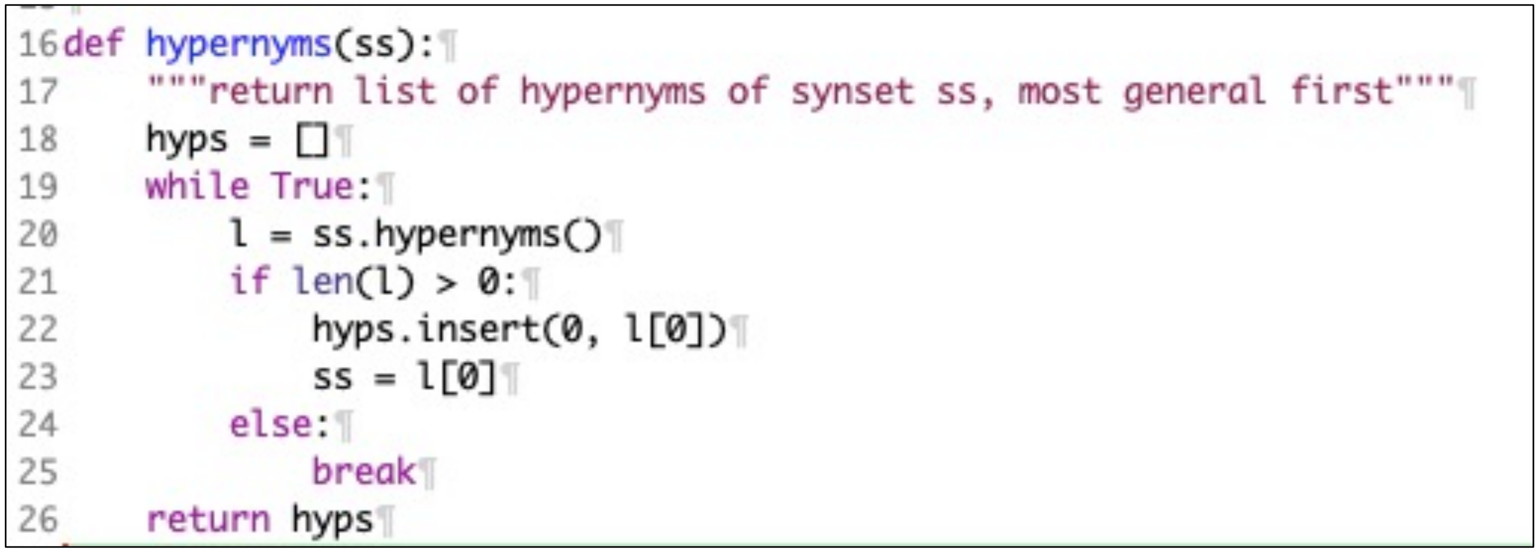
The noun stool has 4 senses (first 1 from tagged texts)

1. (3) stool#1 -- (a simple seat without a back or arms)
2. fecal matter#1, faecal matter#1, feces#1, faeces#1, BM#1, stool#2, ordure#1, dejection#2 -- (solid excretory product evacuated from the bowels)
3. stool#3 -- ((forestry) the stump of a tree that has been felled or headed for the production of saplings)
4. toilet#2, can#5, commode#1, crapper#1, pot#2, potty#1, stool#4, throne#2 -- (a plumbing fixture for defecation and urination)

The verb stool has 4 senses (no senses from tagged texts)

1. stool#1 -- (lure with a stool, as of wild fowl)
2. stool#2 -- (react to a decoy, of wildfowl)
3. stool#3, tiller#1 -- (grow shoots in the form of stools or tillers)
4. stool#4, defecate#1, shit#2, take a shit#1, take a crap#1, ca-ca#1, crap#1, make#29 -- (have a bowel movement; "The dog had made in the flower beds")

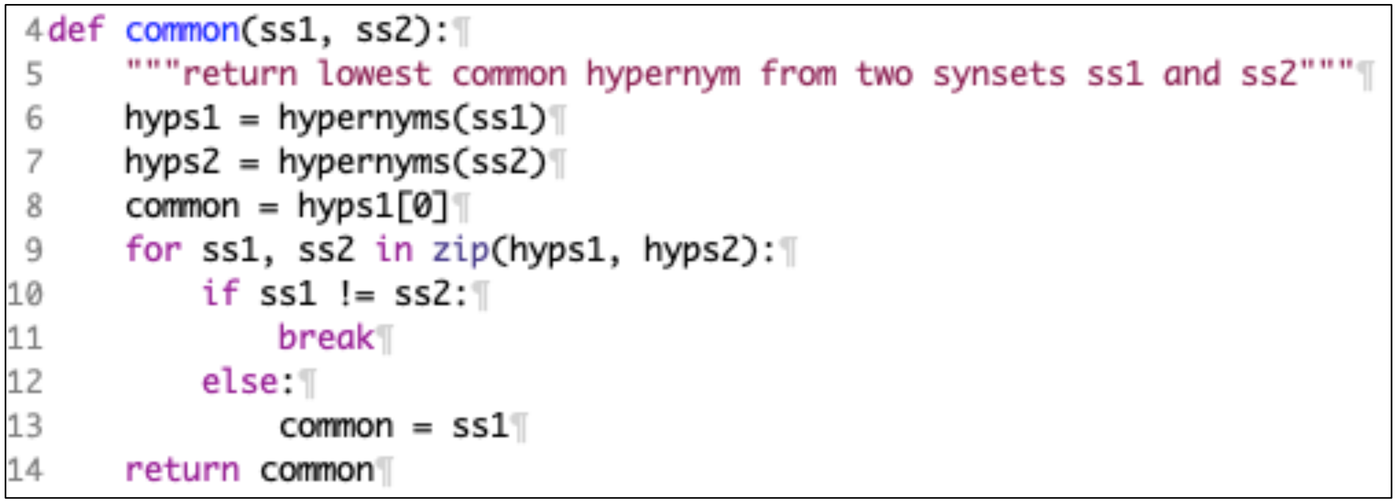
Lowest common hypernym



```python
def hypernyms(ss):
    """return list of hypernyms of synset ss, most general first"""
    hyps = []
    while True:
        l = ss.hypernyms()
        if len(l) > 0:
            hyps.insert(0, l[0])
            ss = l[0]
        else:
            break
    return hyps
```

```
>> hypernyms(wn.synset('table.n.02'))
[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')]
```

Lowest common hypernym

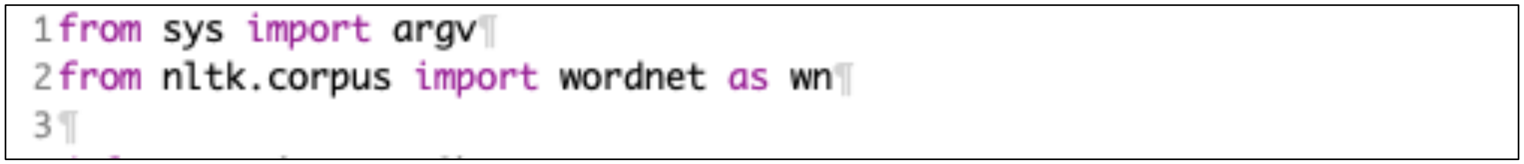


```python
def common(ss1, ss2):
    """return lowest common hypernym from two synsets ss1 and ss2"""
    hyps1 = hypernyms(ss1)
    hyps2 = hypernyms(ss2)
    common = hyps1[0]
    for ss1, ss2 in zip(hyps1, hyps2):
        if ss1 != ss2:
            break
        else:
            common = ss1
    return common
```

```
>>> hypernyms(wn.synset('table.n.02'))
[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')]
>>> hypernyms(wn.synset('stool.n.01'))
[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')]
```

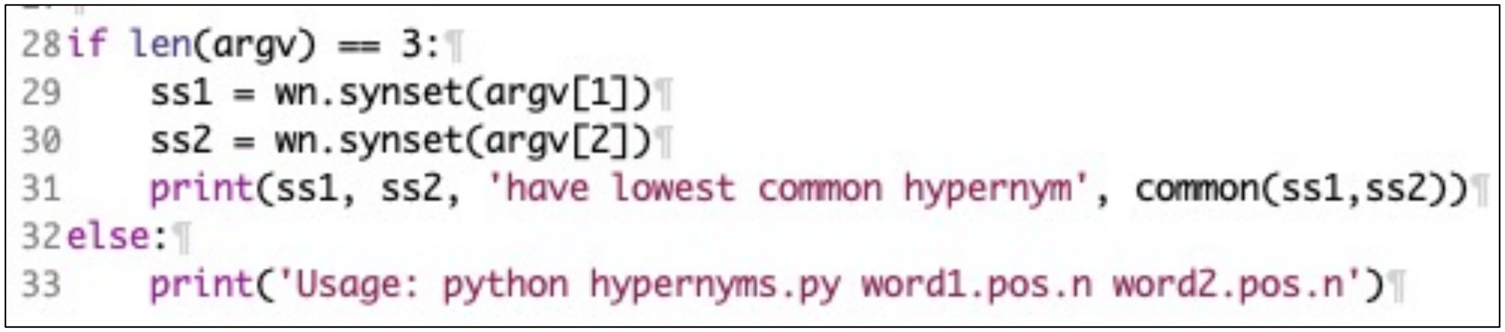
Lowest common hypernym

hypernyms.py



```python
from sys import argv
from nltk.corpus import wordnet as wn

```



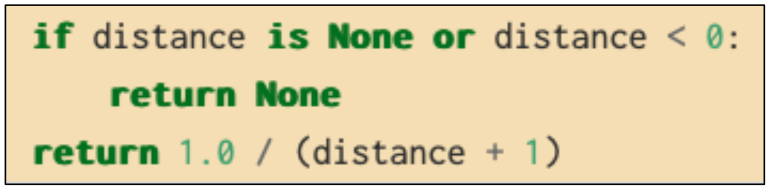
```python
if len(argv) == 3:
    ss1 = wn.synset(argv[1])
    ss2 = wn.synset(argv[2])
    print(ss1, ss2, 'have lowest common hypernym', common(ss1,ss2))
else:
    print('Usage: python hypernyms.py word1.pos.n word2.pos.n')
```

Lowest common hypernym

- Examples:
  - `python hypernyms.py table.n.02 stool.n.01`
    ```
    Synset('table.n.02') Synset('stool.n.01') have lowest common
    hypernym Synset('furniture.n.01')
    ```
  - `python hypernyms.py table.n.04 stool.n.01`
    ```
    Synset('mesa.n.01') Synset('stool.n.01') have lowest common
    hypernym Synset('object.n.01')
    ```
  - `python hypernyms.py table.n.02 stool.n.04`
    ```
    Synset('table.n.02') Synset('toilet.n.02') have lowest common
    hypernym Synset('artifact.n.01')
    ```
    ```
    python hypernyms.py table.n.02 house.n.01
    ```
    ```
    Synset('table.n.02') Synset('house.n.01') have lowest common
    hypernym Synset('artifact.n.01')
    ```

.path_similarity()

- https://www.nltk.org/_modules/nltk/corpus/reader/wordnet.html#Synset.path_similarity



```
if distance is None or distance < 0:
    return None
return 1.0 / (distance + 1)
```

```
Other similarity metrics:
.lch_similarity()
.wup_similarity()
.res_similarity()
.jcn_similarity()
.lin_similarity()
```

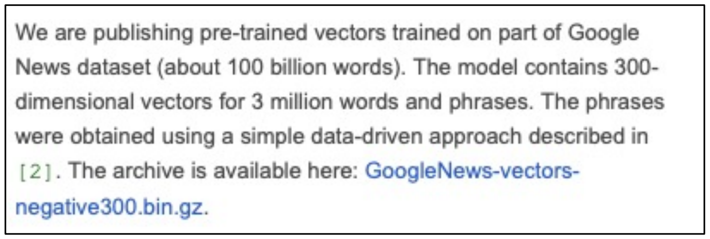
- Examples:
  - `wn.synset('table.n.02').path_similarity(wn.synset('stool.n.01'))`
    `0.25`
  - `wn.synset('table.n.02').path_similarity(wn.synset('seat.n.03'))`
    `0.3333333333333333`
  - `wn.synset('furniture.n.01').path_similarity(wn.synset('seat.n.03'))`
    `0.5`
  - `wn.synset('table.n.02').path_similarity(wn.synset('table.n.02'))`
    `1.0`

Word Embeddings: similarity

- Install gensim
  - Gensim is a Python library for *topic modelling*, *document indexing* and *similarity retrieval* with large corpora.
  - Target audience is the *natural language processing* (NLP) and *information retrieval* (IR) community.
  - **Note**: assumes numpy is installed
  - https://radimrehurek.com/gensim/
  - `conda install -c conda-forge gensim`
  - `pip3 install --upgrade gensim`
- Use with nltk:
  - https://www.nltk.org/howto/gensim.html

Word Embeddings: similarity

- NLTK includes a pre-trained model which is part of a model that is trained on 100 billion words from the Google News Dataset.
  - https://code.google.com/archive/p/word2vec/


We are publishing pre-trained vectors trained on part of Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were obtained using a simple data-driven approach described in [2]. The archive is available here: GoogleNews-vectors-negative300.bin.gz.

- python

```
>>> from nltk.data import find
>>> word2vec_sample = str(find('models/word2vec_sample/pruned.word2vec.txt'))
>>> import gensim
>>> model = gensim.models.KeyedVectors.load_word2vec_format(word2vec_sample,
binary=False)
>>> len(model)
43981
>>> len(model['stool'])
300
```

Word Embeddings: similarity

```
>>> len(model)
43981
>>> len(model['stool'])
300
>>> model.most_similar(positive=['stool'], topn=10)
[('worktable', 0.4907768666744232), ('sofa', 0.4850592315196991),
('footstool', 0.4750695526599884), ('couch', 0.44956985116004944),
('mat', 0.4150840938091278), ('banister', 0.41452112793922424),
('chair', 0.4106588363647461), ('uncurled', 0.4053769111633301),
('chaise', 0.40421634912490845), ('hammock', 0.39813506603240967)]
```

WordNet: coordinate terms (sister term)

4 senses of stool

Sense 1

stool#1 -- (a simple seat without a back or arms)

-> seat#3 -- (furniture that is designed for sitting on; "there were not enough seats for all the guests")

=> bench#1 -- (a long seat for more than one person)

=> bench#7 -- ((law) the seat for judges in a courtroom)

=> box#8, box seat#2 -- (the driver's seat on a coach; "an armed guard sat in the box with the driver")

=> box seat#1 -- (a special seat in a theater or grandstand box)

=> chair#1 -- (a seat for one person, with a support for the back; "he put his coat over the back of the chair and sat down")

=> ottoman#3, pouf#2, pouffe#1, puff#6, hassock#1 -- (thick cushion used as a seat)

HAS INSTANCE=> Siege Perilous#1 -- (the legendary seat at King Arthur's Round Table reserved for the knight who would find the Holy Grail; it was fatal for anyone else to sit in it)

=> sofa#1, couch#1, lounge#1 -- (an upholstered seat for more than one person)

=> stool#1 -- (a simple seat without a back or arms)

=> toilet seat#1 -- (the hinged seat on a toilet)

WordNet: hyponyms (full)

3 of 4 senses of stool

Sense 1

stool#1 -- (a simple seat without a back or arms)

=> campstool#1 -- (a folding stool)

=> cutty stool#1 -- (a low stool; formerly in Scotland, a seat in a church where an offender was publicly rebuked)

=> footstool#1, footrest#1, ottoman#4, tuffet#1 -- (a low seat or a stool to rest the feet of a seated person)

=> milking stool#1 -- (low three-legged stool with a half round seat; used to sit on while milking a cow)

=> music stool#1, piano stool#1 -- (a stool for piano players; usually adjustable in height)

=> step stool#1 -- (a stool that has one or two steps that fold under the seat)

=> taboret#1, tabouret#1 -- (a low stool in the shape of a drum)

Case: hammock and stool

The noun *hammock* has 2 senses (no senses from tagged texts)

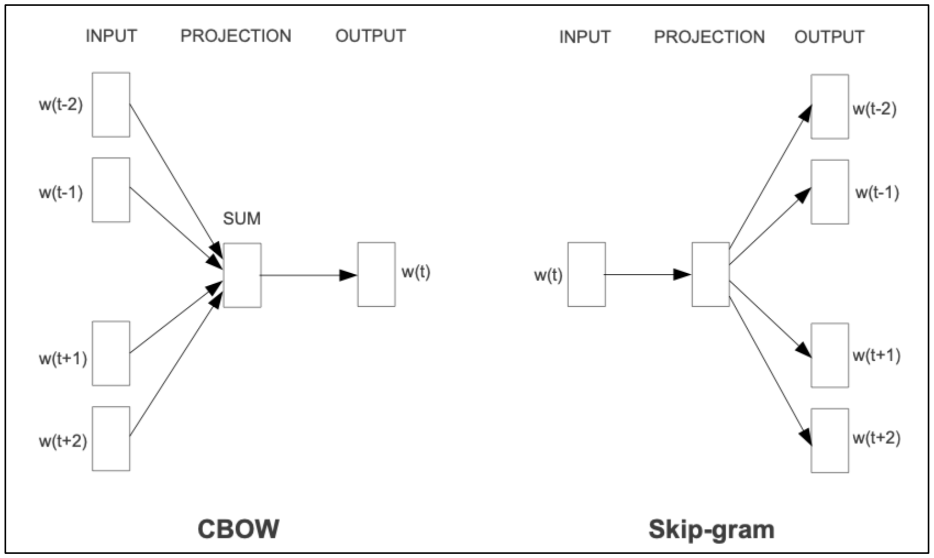
- 1. knoll#1, mound#2, hillock#1, hummock#1, hammock#1 -- (a small natural hill)
- 2. hammock#2, sack#6 -- (a hanging bed of canvas or rope netting (usually suspended between two trees); swings easily)

- WordNet:
  - `from nltk.corpus import wordnet as wn`
  - `wn.synset('stool.n.01').path_similarity(wn.synset('hammock.n.02'))`
  - `0.16666666666666666`

  - `python hypernyms.py hammock.n.02 stool.n.01`
  - `Synset('hammock.n.02') Synset('stool.n.01') have lowest common hypernym Synset('furniture.n.01')`

Word Embeddings: similarity

- word2vec (Mikolov et al., 2013): *representing words as vectors*
- 2-layer neural net model trained on large corpora
- more advanced models: e.g. GloVe, BERT

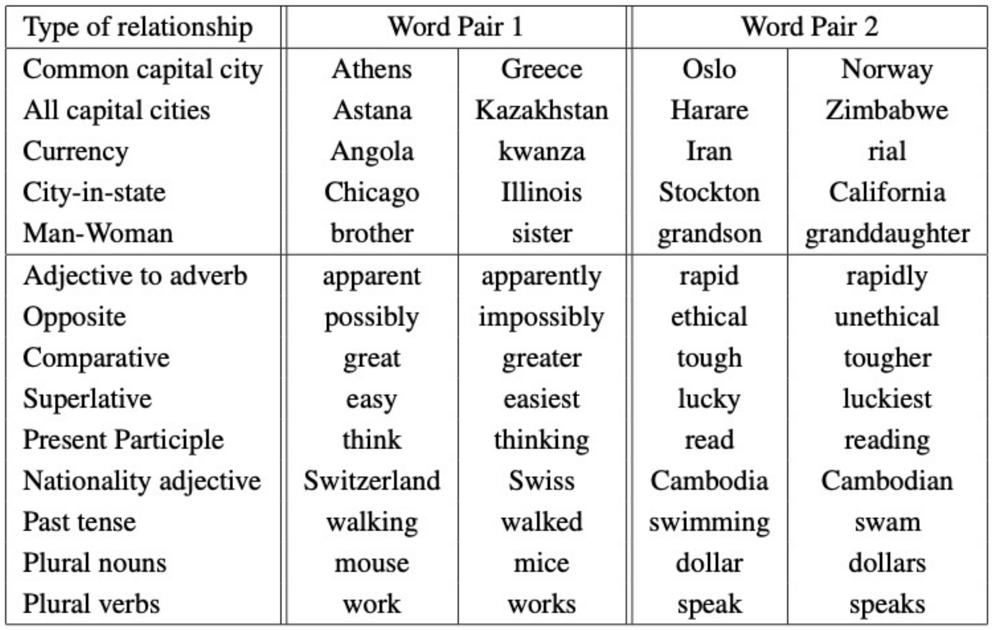


In models using large corpora and a high number of dimensions, the skip-gram model yields the highest overall accuracy, and consistently produces the highest accuracy on semantic relationships, as well as yielding the highest syntactic accuracy in most cases. However, the CBOW is less computationally expensive and yields similar accuracy results. (Wikipedia)

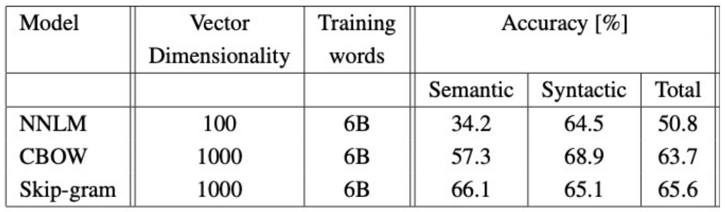
CBOW = Continuous Bag of Words

Word Embeddings: similarity

Table 1: *Examples of five types of semantic and nine types of syntactic questions in the Semantic-Syntactic Word Relationship test set.*



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks



Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
NNLM	100	6B	34.2	64.5	50.8
CBOW	1000	6B	57.3	68.9	63.7
Skip-gram	1000	6B	66.1	65.1	65.6

.most_similar()

- Semantic Examples:
  - the vector ‘King - Man + Woman’ is close to ‘Queen’

```
>>> model.most_similar(positive=['woman','king'],
negative=['man'], topn = 3)
[('queen', 0.7118193507194519), ('monarch', 0.6189674139022827),
('princess', 0.5902430415153503)]
```

  - ‘Germany - Berlin + Paris’ is close to ‘France’.

```
>>> model.most_similar(positive=['Paris','Germany'],
negative=['Berlin'], topn = 3)
[('France', 0.7884091138839722), ('Belgium',
0.6197876930236816), ('Spain', 0.5664774179458618)]
```

.most_similar()

- Semantic Examples:
  - granddaughter – sister + brother maps to grandson

```
>>> model.most_similar(positive=['brother','granddaughter'],
negative=['sister'], topn = 3)
[('grandson', 0.8567124605178833), ('nephew',
0.8048675656318665), ('son', 0.8035288453102112)]
```

  - grandmother – sister + brother maps to uncle

```
>>> model.most_similar(positive=['brother','grandmother'],
negative=['sister'], topn = 3)
[('uncle', 0.7744704484939575), ('father', 0.7595500349998474),
('grandfather', 0.7549421787261963)]
```

.most_similar()

- Syntactic Examples:
  - bigger - small + big maps to biggest

```
>>> model.most_similar(positive=['big','bigger'], negative=['small'], topn = 3)
[('biggest', 0.5590152740478516), ('huge', 0.5318294167518616), ('helluva', 0.4900902211666107)]
```

  - quicker - slow + quick maps to quickest

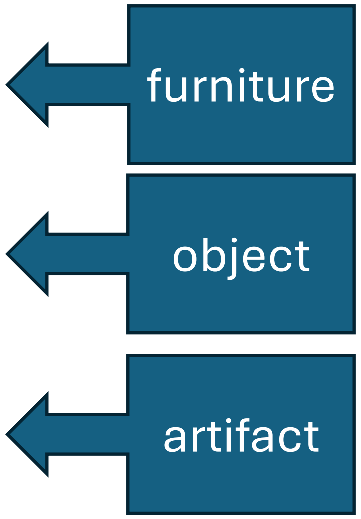
```
>>> model.most_similar(positive=['quick','quicker'], negative=['slow'], topn = 3)
[('quickest', 0.48321864008903503), ('faster', 0.4744618833065033), ('easier', 0.4644150733947754)]
```

  - colder - warm + cold maps to warmer

```
>>> model.most_similar(positive=['cold','colder'], negative=['warm'], topn = 3)
[('warmer', 0.5285316109657288), ('Cold', 0.5240233540534973), ('frigid', 0.5069639086723328)]
```

Word Embeddings: similarity

```
>>> model.similarity('table','stool')
0.37125778
>>> model.similarity('mesa','stool')
KeyError: "Key 'mesa' not present"
>>> model.similarity('table','toilet')
0.20692933
>>> model.similarity('table','house')
0.19165623
```



The noun mesa has 2 senses (no senses from tagged texts)
1. mesa#1, table#4 -- (flat tableland with steep edges; "the tribe was relatively safe on the mesa but they had to descend into the valley for water")
2. Mesa#2 -- (a city in Arizona just to the east of Phoenix; originally a suburb of Phoenix)