LING/C SC 581:
Advanced Computational Linguistics

Lecture 18

Prof. Sandiway Fong

Today's Topic

- Homework 9 Review
- A look at ***similarity***
  - WordNet vs. Word Embeddings

Homework 9 Review

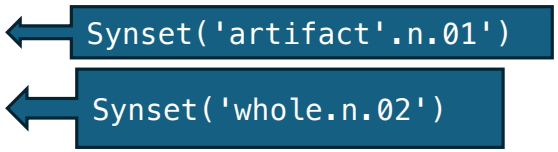
- Q1: *automobile* (46) vs. *aeroplane* (25) vs. *motorcycle* (3)

```
>>> list(wn.synset('automobile.n.01').closure(lambda x:x.part_meronyms()))
[Synset('air_bag.n.01'), Synset('hood.n.09'), Synset('luggage_compartment.n.01'), Synset('automobile_engine.n.01'),
  Synset('roof.n.02'), Synset('auto_accessory.n.01'), Synset('gasoline_engine.n.01'), Synset('sunroof.n.01'),
  Synset('automobile_horn.n.01'), Synset('buffer.n.06'), Synset('fender.n.01'), Synset('rear_window.n.01'),
  Synset('floorboard.n.02'), Synset('glove_compartment.n.01'), Synset('grille.n.02'), Synset('car_window.n.01'),
  Synset('accelerator.n.01'), Synset('first_gear.n.01'), Synset('car_mirror.n.01'), Synset('stabilizer_bar.n.01'),
  Synset('car_door.n.01'), Synset('reverse.n.02'), Synset('bumper.n.02'), Synset('car_seat.n.01'),
  Synset('high_gear.n.01'), Synset('third_gear.n.01'), Synset('window.n.02'), Synset('tail_fin.n.02'),
  Synset('running_board.n.01'), Synset('hood_ornament.n.01'), Synset('exhaust.n.02'),
  Synset('inlet_manifold.n.01'), Synset('horn_button.n.01'), Synset('hinge.n.01'), Synset('armrest.n.01'),
  Synset('doorlock.n.01'), Synset('bumper_guard.n.01'), Synset('back.n.08'), Synset('headrest.n.01'),
  Synset('seat_belt.n.01'), Synset('exhaust_pipe.n.01'), Synset('exhaust_manifold.n.01'), Synset('tailpipe.n.01'),
  Synset('silencer.n.02'), Synset('exhaust_valve.n.01'), Synset('pintle.n.01')]

>>> list(wn.synset('airplane.n.01').closure(lambda x:x.part_meronyms()))
[Synset('windshield.n.01'), Synset('accelerator.n.01'), Synset('pod.n.04'), Synset('landing_gear.n.01'),
  Synset('radome.n.01'), Synset('wing.n.02'), Synset('hood.n.09'), Synset('navigation_light.n.01'),
  Synset('fuselage.n.01'), Synset('escape_hatch.n.01'), Synset('nosewheel.n.01'), Synset('rib.n.01'),
  Synset('aileron.n.01'), Synset('flap.n.05'), Synset('hood_ornament.n.01'), Synset('porthole.n.01'),
  Synset('tail.n.07'), Synset('deadlight.n.01'), Synset('horizontal_tail.n.01'), Synset('vertical_tail.n.01'),
  Synset('stabilizer.n.02'), Synset('horizontal_stabilizer.n.01'), Synset('elevator.n.02'), Synset('rudder.n.01'),
  Synset('vertical_stabilizer.n.01')]

>>> list(wn.synset('motorcycle.n.01').closure(lambda x:x.part_meronyms()))
[Synset('kick_starter.n.01'), Synset('kickstand.n.01'), Synset('mudguard.n.01')]
```

Homework 9 Review

Question 2: part meronym inheritance

```
>>> for ss in wn.synset('armchair.n.01').closure(lambda x:x.hypernyms()):
...     print(ss.part_meronyms())
...
[Synset('back.n.08'), Synset('leg.n.03')]
[Synset('upholstery.n.01'), Synset('seat.n.04')]
[]
[]
[]
[]
[Synset('section.n.04'), Synset('part.n.02')]
[]
[]
[]
```



Homework 9 Review

- Question 3: items with *arms*

```
>>> wn.synset('arm.n.01').part_holonyms()
[Synset('body.n.01'), Synset('homo.n.02')]
>>> for ss in wn.synset('arm.n.01').part_holonyms():
...     print(list(ss.closure(lambda x:x.hyponyms())))
...
[Synset('life_form.n.01'), Synset('live_body.n.01'), Synset('human_body.n.01'),
  Synset('system.n.08'), Synset('adult_body.n.01'), Synset('juvenile_body.n.01'),
  Synset('male_body.n.01'), Synset('female_body.n.01'), Synset('person.n.02'),
  Synset('adult_male_body.n.01'), Synset('adult_female_body.n.01'),
  Synset('child's_body.n.01')]
[Synset('homo_habilis.n.01'), Synset('homo_soloensis.n.01'),
  Synset('neandertal_man.n.01'), Synset('rhodesian_man.n.01'), Synset('world.n.08'),
  Synset('homo_sapiens.n.01'), Synset('homo_erectus.n.01'), Synset('solo_man.n.01'),
  Synset('homo_sapiens_sapiens.n.01'), Synset('cro-magnon.n.01'),
  Synset('boskop_man.n.01'), Synset('java_man.n.01'), Synset('peking_man.n.01')]
>>> wn.synset('world.n.08').definition()
'all of the living human inhabitants of the earth'
```

Homework 9 Review

- Question 3: items with *arms*

```
>>> for ss in wn.synset('arm.n.04').part_holonyms():
...     print(list(ss.closure(lambda x:x.hyponyms())))
...
[Synset('easy_chair.n.01'), Synset('recliner.n.01'),
Synset('captain's_chair.n.01'), Synset('morris_chair.n.01'),
Synset('fauteuil.n.01'), Synset('wing_chair.n.01')]
```

Homework 9 Review

- Question 3: items with *arms*

```
>>> s = set()
>>> for ss in wn.synset('arm.n.06').part_holonyms():
...     s.update(set(ss.closure(lambda x:x.hyponyms())))
...
>>> len(s)
271
```

Homework 9 Review

- Question 3: items with *arms*

```
>>> for ss in wn.synset('arm.n.06').part_holonyms():
...     print(list(ss.closure(lambda x:x.hyponyms())))
```

```
[Synset('peplos.n.01'), Synset('silks.n.01'), Synset('haik.n.01'), Synset('jump_suit.n.01'), Synset('reversible.n.01'), Synset('hose.n.02'), Synset('robe.n.01'), Synset('romper.n.02'),
    Synset('shirt.n.01'), Synset('sweater.n.01'), Synset('burqa.n.01'), Synset('overgarment.n.01'), Synset('weeds.n.01'), Synset('scarf.n.01'), Synset('sweat_suit.n.01'),
    Synset('leotard.n.01'), Synset('straitjacket.n.02'), Synset('undergarment.n.01'), Synset('stomacher.n.01'), Synset('hand-me-down.n.01'), Synset('separate.n.02'), Synset('fur.n.03'),
    Synset('sealskin.n.02'), Synset('wet_suit.n.01'), Synset('breechcloth.n.01'), Synset('trouser.n.02'), Synset('neckwear.n.01'), Synset('kanzu.n.01'), Synset('swimsuit.n.01'),
    Synset('head_covering.n.01'), Synset('camlet.n.01'), Synset('vest.n.01'), Synset('legging.n.01'), Synset('mending.n.01'), Synset('raglan.n.01'), Synset('skirt.n.02'), Synset('suit.n.01'),
    Synset('swaddling_clothes.n.01'), Synset('trouser.n.01'), Synset('ironing.n.01'), Synset('sackcloth.n.01'), Synset('wraparound.n.01'), Synset('gown.n.04'), Synset('laundry.n.01'),
    Synset('sunsuit.n.01'), Synset('scapular.n.02'), Synset('motley.n.02'), Synset('diaper.n.01'), Synset('dressing_gown.n.01'), Synset('abaya.n.01'), Synset('kimono.n.01'),
    Synset('bathrobe.n.01'), Synset('polo_shirt.n.01'), Synset('dress_shirt.n.01'), Synset('dashiki.n.01'), Synset('camise.n.01'), Synset('work_shirt.n.01'), Synset('jersey.n.03'),
    Synset('hair_shirt.n.01'), Synset('tank_top.n.01'), Synset('kurta.n.01'), Synset('cardigan.n.01'), Synset('pullover.n.01'), Synset('turtleneck.n.01'), Synset('izar.n.01'),
    Synset('aba.n.01'), Synset('cloak.n.02'), Synset('coat.n.01'), Synset('snowsuit.n.01'), Synset('muffler.n.02'), Synset('kerchief.n.01'), Synset('rebozo.n.01'), Synset('sable.n.04'),
    Synset('stole.n.01'), Synset('feather_boa.n.01'), Synset('fichu.n.01'), Synset('lambrequin.n.01'), Synset('tudung.n.01'), Synset('mantilla.n.01'), Synset('patka.n.01'),
    Synset('brassiere.n.01'), Synset('foundation_garment.n.01'), Synset('singlet.n.01'), Synset('garter_belt.n.01'), Synset('chemise.n.01'), Synset('petticoat.n.01'),
    Synset('underwear.n.01'), Synset('body_stocking.n.01'), Synset('long_underwear.n.01'), Synset('underpants.n.01'), Synset('camisole.n.01'), Synset('dhoti.n.01'), Synset('necktie.n.01'),
    Synset('cravat.n.01'), Synset('swimming_trunks.n.01'), Synset('maillot.n.01'), Synset('bikini.n.02'), Synset('yashmak.n.01'), Synset('face_veil.n.01'), Synset('chador.n.01'),
    Synset('bulletproof_vest.n.01'), Synset('spat.n.02'), Synset('chap.n.04'), Synset('puttee.n.01'), Synset('gaiter.n.03'), Synset('overskirt.n.01'), Synset('miniskirt.n.01'),
    Synset('ballet_skirt.n.01'), Synset('sarong.n.01'), Synset('grass_skirt.n.01'), Synset('iful_skirt.n.01'), Synset('maxi.n.01'), Synset('gathered_skirt.n.01'), Synset('hoopskirt.n.01'),
    Synset('hobble_skirt.n.01'), Synset('lavalava.n.01'), Synset('culottes.n.01'), Synset('kilt.n.01'), Synset('single-breasted_suit.n.01'), Synset('slack_suit.n.01'), Synset('double-
    breasted_suit.n.01'), Synset('zoot_suit.n.01'), Synset('business_suit.n.01'), Synset('pinstripe.n.01'), Synset('churidars.n.01'), Synset('slacks.n.01'), Synset('chino.n.01'),
    Synset('jodhpurs.n.01'), Synset('jeap.n.01'), Synset('long_trousers.n.01'), Synset('pantaloon.n.03'), Synset('short_pants.n.01'), Synset('pedal_pusher.n.01'), Synset('trews.n.01'),
    Synset('bellbottom_trousers.n.01'), Synset('pajama.n.01'), Synset('salwar.n.01'), Synset('stretch_pants.n.01'), Synset('breeches.n.01'), Synset('flannel.n.03'),
    Synset('sweat_pants.n.01'), Synset('cords.n.01'), Synset('flatwork.n.01'), Synset('sweatshirt.n.01'), Synset('pallium.n.04'), Synset('capote.n.02'), Synset('jellaba.n.01'),
    Synset('poncho.n.01'), Synset('cape.n.02'), Synset('dolman.n.02'), Synset('wrap.n.01'), Synset('tunic.n.02'), Synset('capuchin.n.01'), Synset('shawl.n.01'), Synset('domino.n.02'),
    Synset('burnous.n.01'), Synset('cope.n.02'), Synset('toga.n.01'), Synset('caftan.n.02'), Synset('opera_cloak.n.01'), Synset('box_coat.n.01'), Synset('jacket.n.01'),
    Synset('sheepskin_coat.n.01'), Synset('topper.n.05'), Synset('coatee.n.01'), Synset('greatcoat.n.01'), Synset('surcoat.n.01'), Synset('duffel_coat.n.01'), Synset('frock_coat.n.01'),
    Synset('cutaway.n.02'), Synset('lab_coat.n.01'), Synset('raincoat.n.01'), Synset('sack_coat.n.01'), Synset('fur_coat.n.01'), Synset('newmarket.n.01'), Synset('mackinaw.n.01'),
    Synset('headcart.n.01'), Synset('neckerchief.n.01'), Synset('uptight.n.02'), Synset('corset.n.01'), Synset('roll-on.n.02'), Synset('crinoline.n.02'), Synset('skivvies.n.01'),
    Synset('long_johns.n.01'), Synset('bvd.n.01'), Synset('lingerie.n.01'), Synset('drawers.n.01'), Synset('briefs.n.01'), Synset('bikini_pants.n.01'), Synset('thong.n.02'),
    Synset('bloomers.n.01'), Synset('pantie.n.01'), Synset('windsor_tie.n.01'), Synset('four-in-hand.n.01'), Synset('string_tie.n.01'), Synset('bolo_tie.n.01'), Synset('old_school_tie.n.01'),
    Synset('bow_tie.n.01'), Synset('neckcloth.n.01'), Synset('ascot.n.01'), Synset('nigab.n.01'), Synset('dirndl.n.01'), Synset('pants_suit.n.01'), Synset('two-piece.n.01'), Synset('three-
    piece_suit.n.01'), Synset('levi's.n.01'), Synset('lederhosen.n.01'), Synset('bermuda_shorts.n.01'), Synset('hot_pants.n.02'), Synset('britches.n.01'), Synset('trunk_hose.n.01'),
    Synset('plus_fours.n.01'), Synset('buckskins.n.01'), Synset('mantelet.n.02'), Synset('pelisse.n.01'), Synset('tippet.n.01'), Synset('chlamys.n.02'), Synset('chiton.n.01'),
    Synset('kirtle.n.01'), Synset('tabard.n.01'), Synset('gymslip.n.01'), Synset('kameez.n.01'), Synset('surcoat.n.02'), Synset('prayer_shawl.n.01'), Synset('serape.n.01'),
    Synset('toga_virilis.n.01'), Synset('bush_jacket.n.01'), Synset('mao_jacket.n.01'), Synset('jumper.n.06'), Synset('norfolk_jacket.n.01'), Synset('blazer.n.01'), Synset('sack.n.05'),
    Synset('single-breasted_jacket.n.01'), Synset('parka.n.01'), Synset('dolman.n.01'), Synset('banyan.n.02'), Synset('jerkin.n.01'), Synset('donkey_jacket.n.01'),
    Synset('dressing_sack.n.01'), Synset('lumberjack.n.02'), Synset('hug-me-tight.n.01'), Synset('bolero.n.02'), Synset('mess_jacket.n.01'), Synset('double-breasted_jacket.n.01'),
    Synset('bed_jacket.n.01'), Synset('pea_jacket.n.01'), Synset('doublet.n.01'), Synset('bomber_jacket.n.01'), Synset('eton_jacket.n.01'), Synset('swallow-tailed_coat.n.01'),
    Synset('capote.n.01'), Synset('surtout.n.01'), Synset('ulster.n.02'), Synset('chesterfield.n.03'), Synset('prince_albert.n.02'), Synset('trench_coat.n.01'), Synset('macintosh.n.02'),
    Synset('burberry.n.01'), Synset('sable_coat.n.01'), Synset('mink.n.02'), Synset('khimar.n.01'), Synset('babushka.n.01'), Synset('hijab.n.01'), Synset('panty_girdle.n.01'),
    Synset('undies.n.01'), Synset('nightgown.n.01'), Synset('black_tie.n.02'), Synset('white_tie.n.01'), Synset('ski_parka.n.01'), Synset('cagoule.n.01'), Synset('oilskin.n.01')]
```

How do we compute similarity?

- A look at WordNet vs. Word Embeddings

WordNet: *table* and *stool*

Class Exercise: the relation between two senses

- 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")
- 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)

WordNet: *table* and *stool*

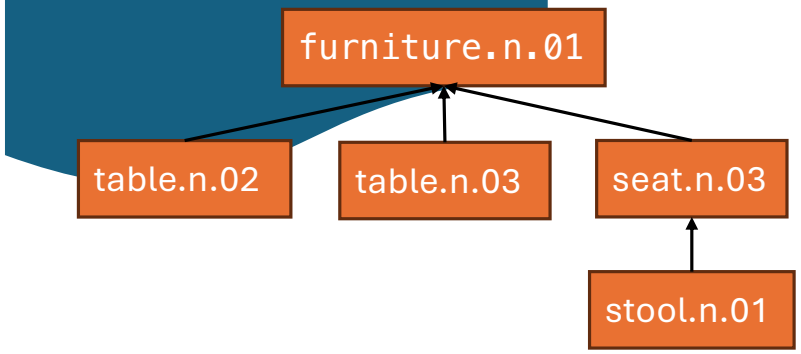
Class Exercise: the relation between two senses

- Consider the following:

```
>>> for ss1 in wn.synsets('table','n'):
...     for ss2 in wn.synsets('stool','n'):
...             print(ss1,ss2,ss1.lowest_common_hypernyms(ss2))
...
```

- *what do you expect to see?*

Lowest common hypernym




```
>>> wn.synset('table.n.02').definition()
'a piece of furniture having a smooth flat top that
is usually supported by one or more vertical legs'
>>> wn.synset('table.n.03').definition()
'a piece of furniture with tableware for a meal
laid out on it'
```

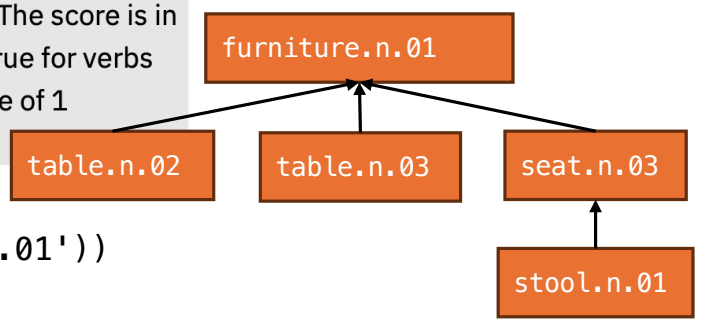
Synset	Synset	common hypernym
'table.n.01'	'stool.n.01'	'entity.n.01'
'table.n.01'	'fecal_matter.n.01'	'abstraction.n.06'
'table.n.01'	'stool.n.03'	'entity.n.01'
'table.n.01'	'toilet.n.02'	'entity.n.01'
'table.n.02'	'stool.n.01'	'furniture.n.01'
'table.n.02'	'fecal_matter.n.01'	'physical_entity.n.01'
'table.n.02'	'stool.n.03'	'whole.n.02'
'table.n.02'	'toilet.n.02'	'artifact.n.01'
'table.n.03'	'stool.n.01'	'furniture.n.01'
'table.n.03'	'fecal_matter.n.01'	'physical_entity.n.01'
'table.n.03'	'stool.n.03'	'whole.n.02'
'table.n.03'	'toilet.n.02'	'artifact.n.01'
'mesa.n.01'	'stool.n.01'	'object.n.01'
'mesa.n.01'	'fecal_matter.n.01'	'physical_entity.n.01'
'mesa.n.01'	'stool.n.03'	'object.n.01'
'mesa.n.01'	'toilet.n.02'	'object.n.01'
'table.n.05'	'stool.n.01'	'entity.n.01'
'table.n.05'	'fecal_matter.n.01'	'abstraction.n.06'
'table.n.05'	'stool.n.03'	'entity.n.01'
'table.n.05'	'toilet.n.02'	'entity.n.01'
'board.n.04'	'stool.n.01'	'physical_entity.n.01'
'board.n.04'	'fecal_matter.n.01'	'matter.n.03'
'board.n.04'	'stool.n.03'	'physical_entity.n.01'
'board.n.04'	'toilet.n.02'	'physical_entity.n.01'

.path_similarity()

path_similarity(*other*, *verbose=False*, *simulate_root=True*)

Path Distance Similarity: Return a score denoting how similar two word senses are, base
shortest path that connects the senses in the is-a (hypernym/hypnoym) taxonomy. The score is in the range 0 to 1, except in those cases where a path cannot be found (will only be true for verbs as there are many distinct verb taxonomies), in which case None is returned. A score of 1 represents identity i.e. comparing a sense with itself will return 1.

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



- **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'))
0.33
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.03'))
0.33
```

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

Word Embeddings: similarity

- Install gensim
  - Gensim is a Python library for *topic modelling*, *document indexing* and *similarity retrieval* with large corpora.
  - https://radimrehurek.com/gensim/
  - `pip3 install 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:

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

#words

word vector length

Word Embeddings: similarity

```
>>> for word,score in model.most_similar(positive=['stool'], topn=10):
...     print(f"{word:10} {score:.2f}")
...
worktable  0.49
sofa       0.49
footstool  0.48
couch      0.45
mat        0.42
banister   0.41
chair      0.41
uncurled   0.41
chaise     0.40
hammock    0.40
```

```
>>> model.similarity('stool','table')
np.float32(0.37125772)
```

Word Embeddings: similarity

stool	gensim	WordNet path_similarity
worktable	0.49	0.2
sofa	0.49	0.33
footstool	0.48	0.5
couch	0.45	0.33
mat	0.42	0.11
banister	0.41	0.1
chair	0.41	0.33
uncurled	0.41	0.07
chaise	0.40	0.25
hammock	0.40	0.17

WordNet: coordinate terms (*sister term*)

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)

0.33 => 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)

0.33 => 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)

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)

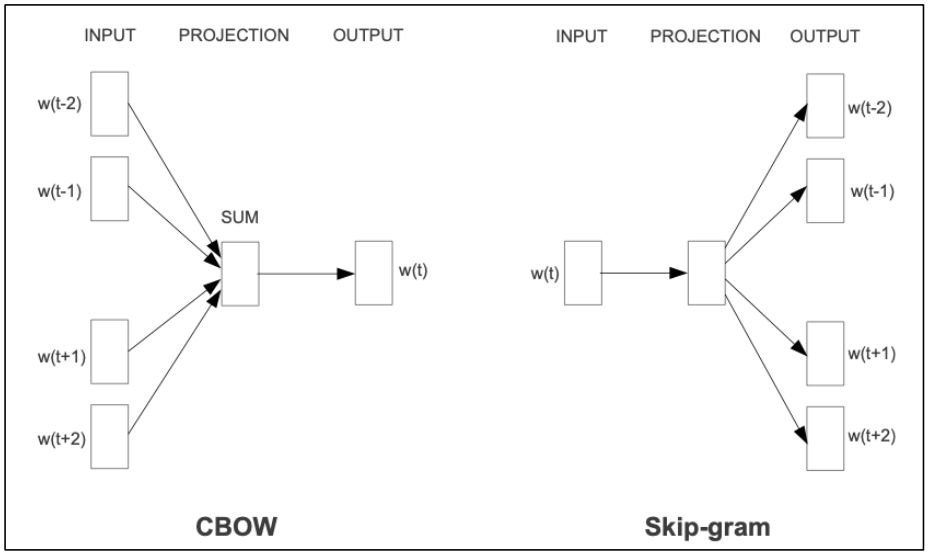
- 1. knoll#1, mound#2, hillock#1, hummock#1, hammock#1 -- (a small natural hill)
- 2. hammock#2, sack#6 -- (a hanging bed of canvas or rope netting (usually suspended between two trees); swings easily)

- WordNet:
  - `from nltk.corpus import wordnet as wn`
  - `wn.synset('stool.n.01').path_similarity(wn.synset('hammock.n.02'))`
  - `0.167`

  - `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

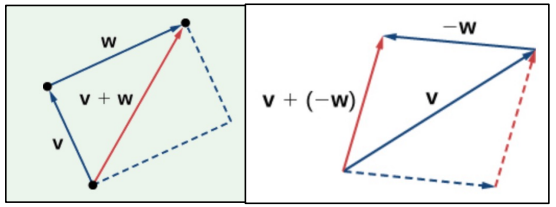
Model	Vector Dimensionality	Training words	Accuracy [%]		
			Semantic	Syntactic	Total
NNLM	100	6B	34.2	64.5	50.8
CBOW	1000	6B	57.3	68.9	63.7
Skip-gram	1000	6B	66.1	65.1	65.6

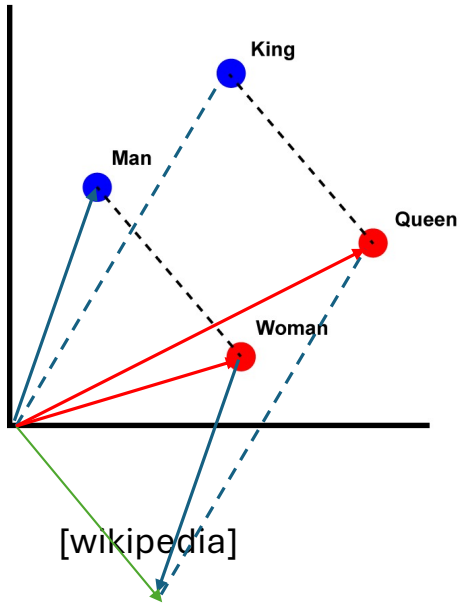
.most_similar()

Vector Arithmetic Semantics (*not possible with* WordNet)

- Example:
- 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)]
```





word2vec

- Examples:
  - https://medium.com/plotly/understanding-word-embedding-arithmetic-why-theres-no-single-answer-to-king-man-woman-cd2760e2cb7f
  - [Y]ou have to include some **'cheating'**. The actual result [is] King - Man + Woman = King. So, the resulting vector would be more similar to King than to Queen. The [...] example only works because [...] the algorithm will exclude the original vector from the possible results! Second comes Queen, which is what the routine will then pick.

- `model.most_similar(positive=['woman','king'], topn=3)`

```
[('man', 0.6628608107566833), ('queen', 0.6438565254211426), ('girl',
0.6136073470115662)]
```

- `model.most_similar(positive=['programmer','woman'],negative=['man'], topn=3)`

```
[('designer', 0.4621824622154236), ('librarian', 0.45112138986587524),
('copywriter', 0.42773979902267456)]
```

- `model.most_similar(positive=['surgeon','woman'],negative=['man'], topn=3)`

```
[('gynecologist', 0.6832519173622131), ('doctor', 0.6073117256164551),
('hysterectomy', 0.5785969495773315)
```

.most_similar()

- Example:
- ‘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 ⟹ grandson

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

- grandmother – sister + brother ⟹ 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 ⟹ biggest

```
>>> model.most_similar(positive=['big','bigger'], negative=['small'], topn = 3)
[('biggest', 0.5590152740478516), ('huge', 0.5318294167518616), ('helluva',
0.490090221166107)]
```

- quicker - slow + quick ⟹ quickest

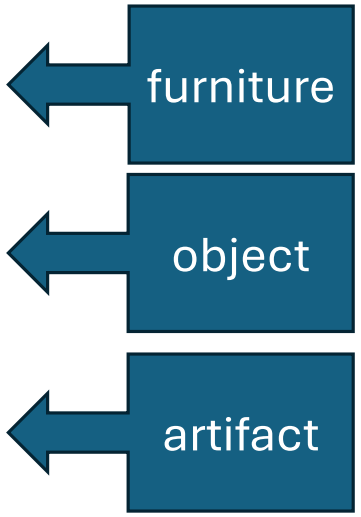
```
>>> model.most_similar(positive=['quick','quicker'], negative=['slow'], topn = 3)
[('quickest', 0.4832186400890503), ('faster', 0.4744618833065033), ('easier',
0.4644150733947754)]
```

- colder - warm + cold ⟹ 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')
0.37
>>> model.similarity('mesa','stool')
KeyError: "Key 'mesa' not present"
>>> model.similarity('table','toilet')
0.21
>>> model.similarity('table','house')
0.19
```



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)