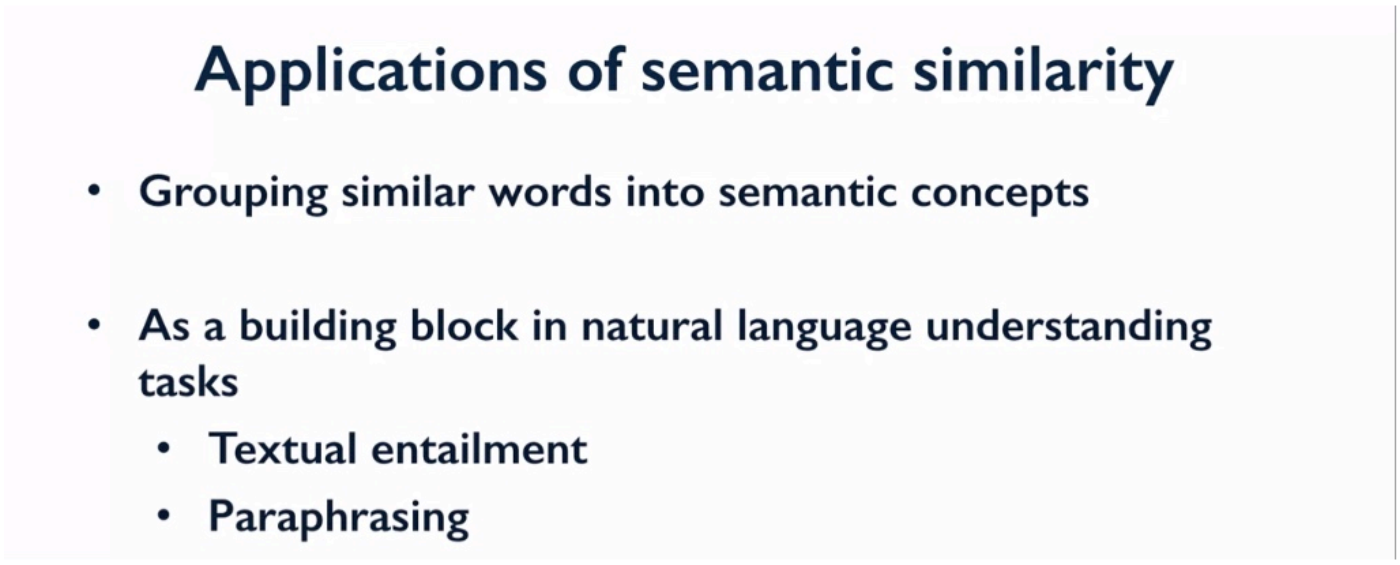
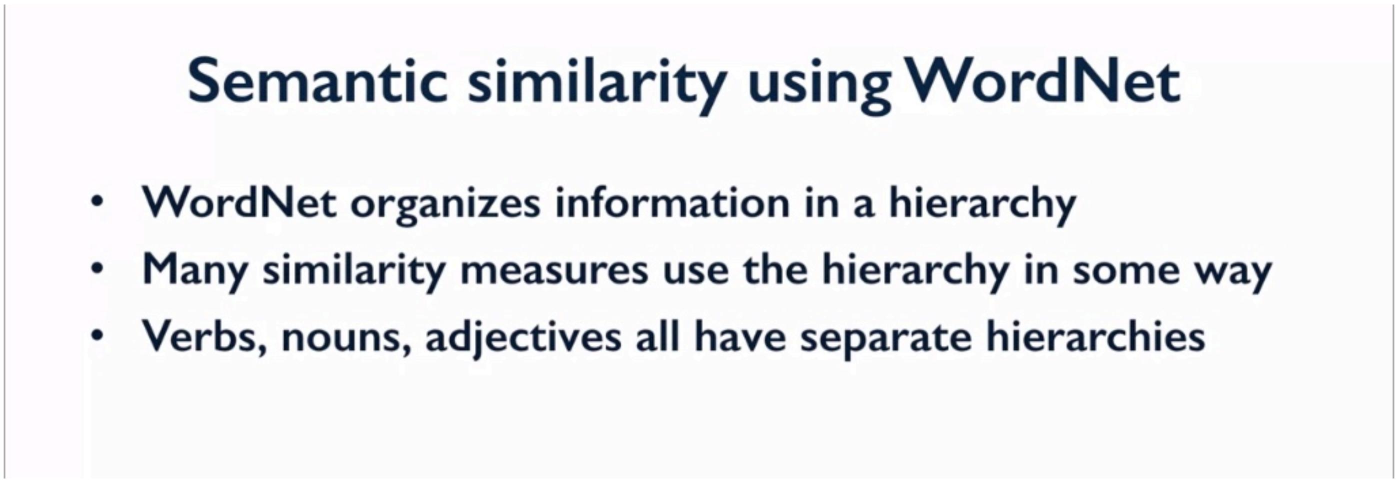
Semantic Text Similarity saved



Applications of semantic similarity

- Grouping similar words into semantic concepts
- As a building block in natural language understanding tasks
  - Textual entailment
  - Paraphrasing

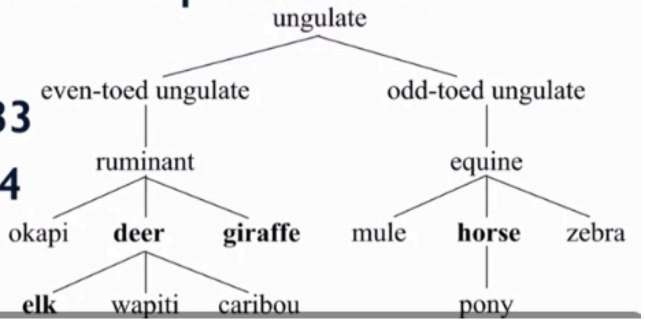


Semantic similarity using WordNet

- WordNet organizes information in a hierarchy
- Many similarity measures use the hierarchy in some way
- Verbs, nouns, adjectives all have separate hierarchies

Path Similarity

- Find the shortest path between the two concepts
- Similarity measure inversely related to path distance
  - PathSim(deer, elk) = 0.5
  - PathSim(deer, giraffe) = 0.33
  - PathSim(deer, horse) = 0.14



Lin Similarity

- Similarity measure based on the information contained in the LCS of the two concepts
  - LinSim(u, v) = 2 x log P(LCS(u,v)) / (log P(u) + log P(v))
- P(u) is given by the information content learnt over a large corpus.

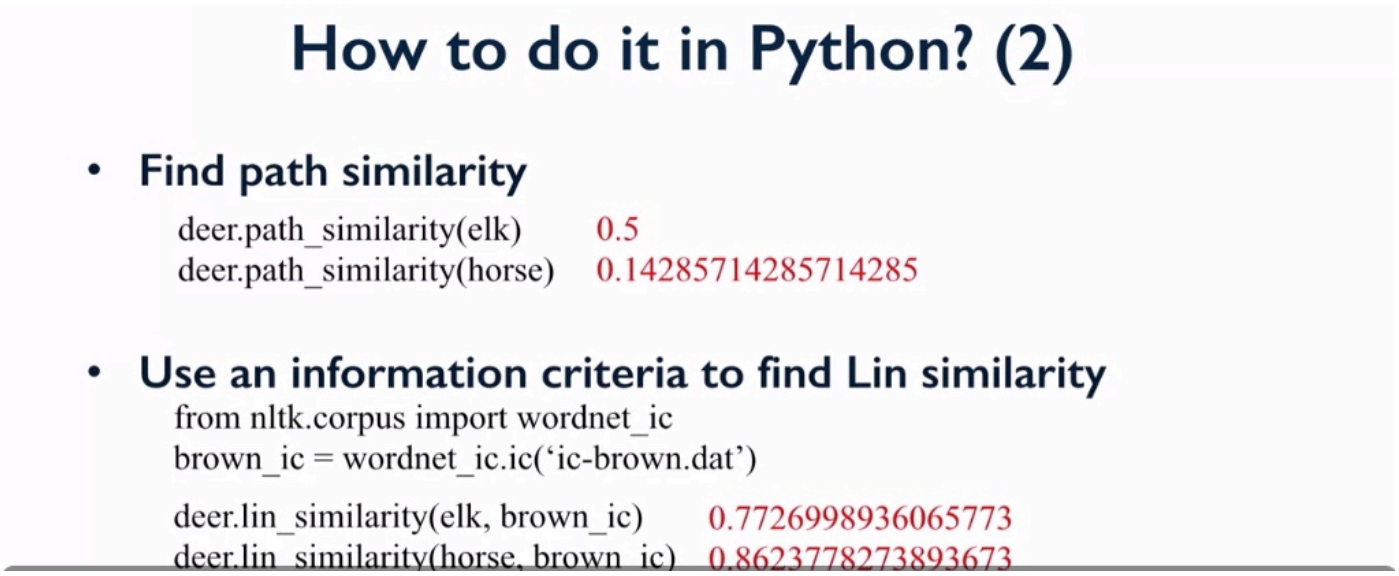
How to do it in Python?

- WordNet easily imported into Python through NLTK

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

- Find appropriate sense of the words

```
deer = wn.synset('deer.n.01')
elk = wn.synset('elk.n.01')
…
```



How to do it in Python? (2)

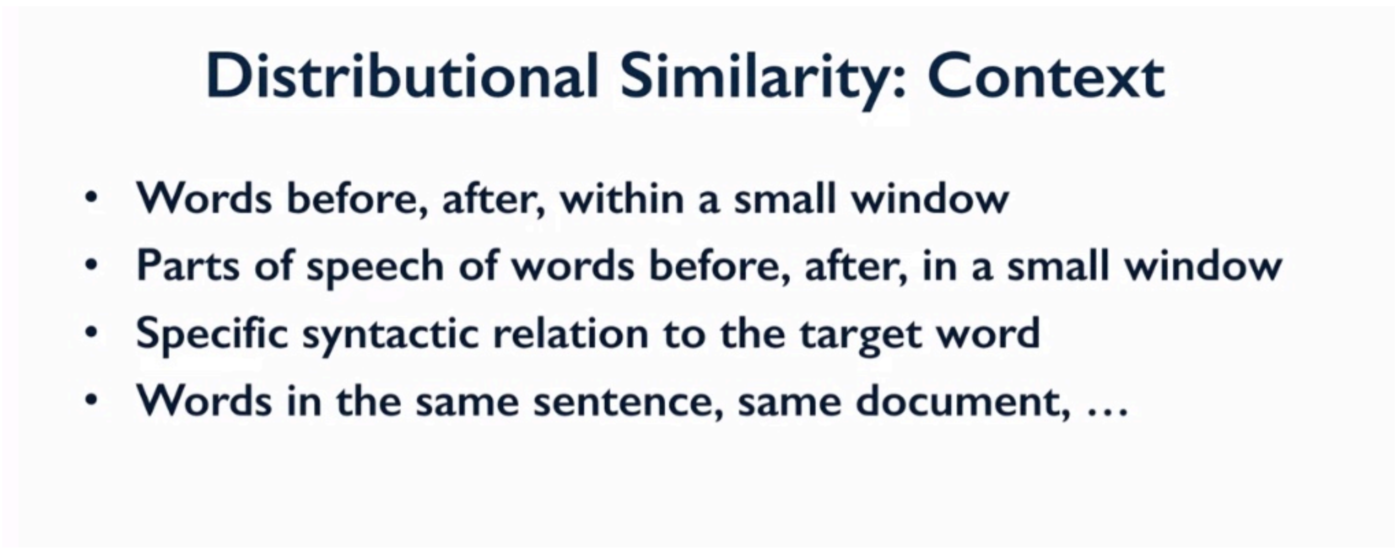
- **Find path similarity**

```
deer.path_similarity(elk)       0.5
deer.path_similarity(horse)     0.14285714285714285
```

- **Use an information criteria to find Lin similarity**

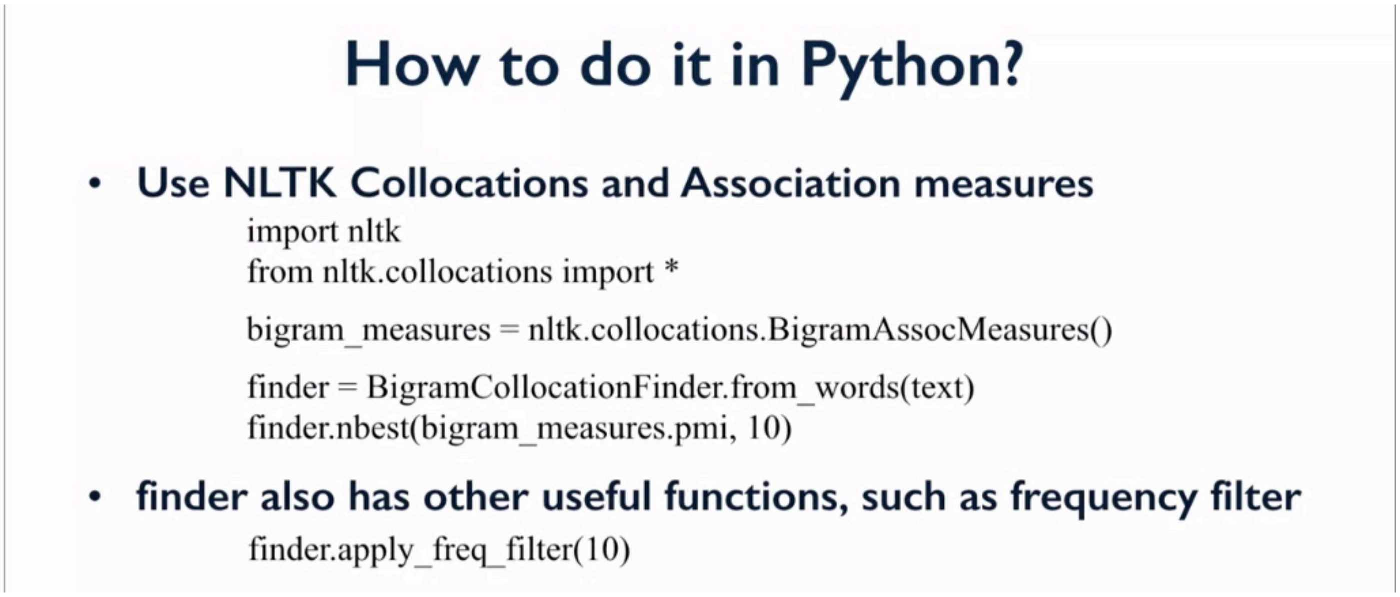
```
from nltk.corpus import wordnet_ic
brown_ic = wordnet_ic.ic('ic-brown.dat')

deer.lin_similarity(elk, brown_ic)       0.7726998936065773
deer.lin_similarity(horse, brown_ic)     0.8623778273893673
```



Distributional Similarity: Context

- **Words before, after, within a small window**
- **Parts of speech of words before, after, in a small window**
- **Specific syntactic relation to the target word**
- **Words in the same sentence, same document, …**



How to do it in Python?

- **Use NLTK Collocations and Association measures**

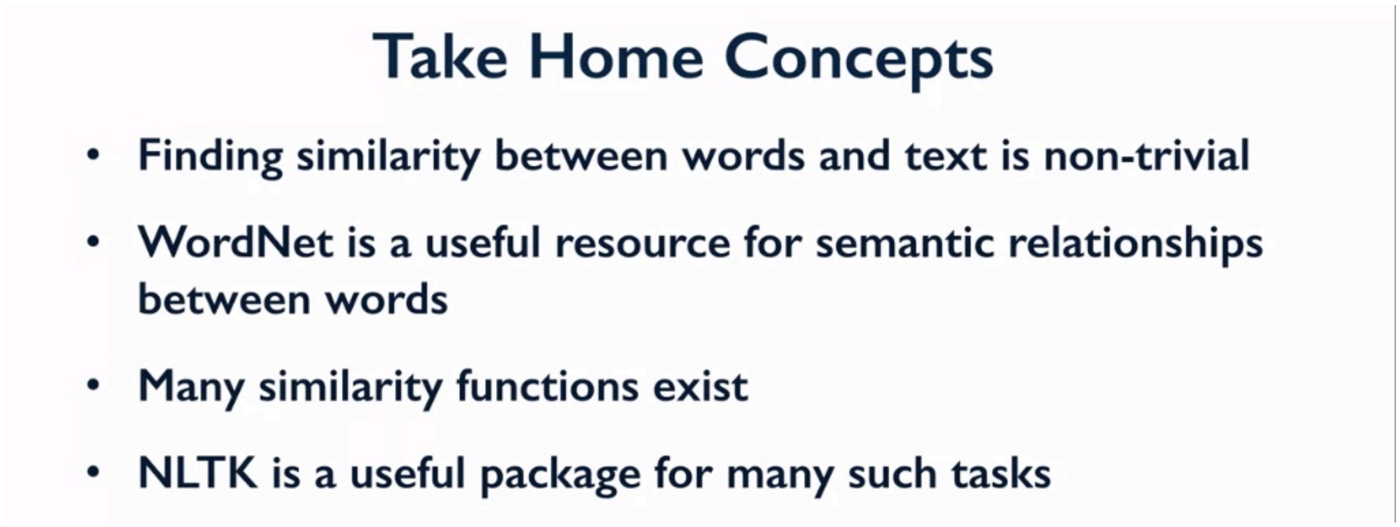
```
import nltk
from nltk.collocations import *

bigram_measures = nltk.collocations.BigramAssocMeasures()

finder = BigramCollocationFinder.from_words(text)
finder.nbest(bigram_measures.pmi, 10)
```

- **finder also has other useful functions, such as frequency filter**

```
finder.apply_freq_filter(10)
```



Take Home Concepts

- Finding similarity between words and text is non-trivial
- WordNet is a useful resource for semantic relationships between words
- Many similarity functions exist
- NLTK is a useful package for many such tasks

Practice Quiz

TOTAL POINTS 2

1. **In the WordNet hierarchy, the word 'puppy' is a direct hyponym of 'dog' (i.e. 'puppy' is a kind of 'dog'. The least common subsumer for 'puppy' and 'dog' is:** 1 point

   - Puppy
   - [✓ Dog]
   - Something other than 'puppy' or 'dog'
   - No least common subsumers exist for hyponym relationships

2. **If 'puppy' is a direct hyponym of 'dog', 'dog' is a direct _____ of 'puppy'** 1 point

   - Hyponym
   - [✓ Hypernym]
   - Meronym
   - Synonym

2. **If the shortest distance between words A and B in the WordNet hierarchy is 6, the path-based similarity measure PathSim(A,B) would be:**

   - 6
   - 1/6 = 0.167
   - 1 - 1/5 = 5/6 = 0.833
   - [✓ 1/(6+1) = 1/7 = 0.143]