Applied Text Mining in Python

Semantic Text Similarity

Which pair of words are most similar?

- deer , elk
- deer , giraffe
- deer , horse
- deer , mouse

Which pair of words are most similar?

- deer , elk
- deer , horse
- deer , house
- deer , roof

- How can we quantify such similarity?

Applications of Text Similarity

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

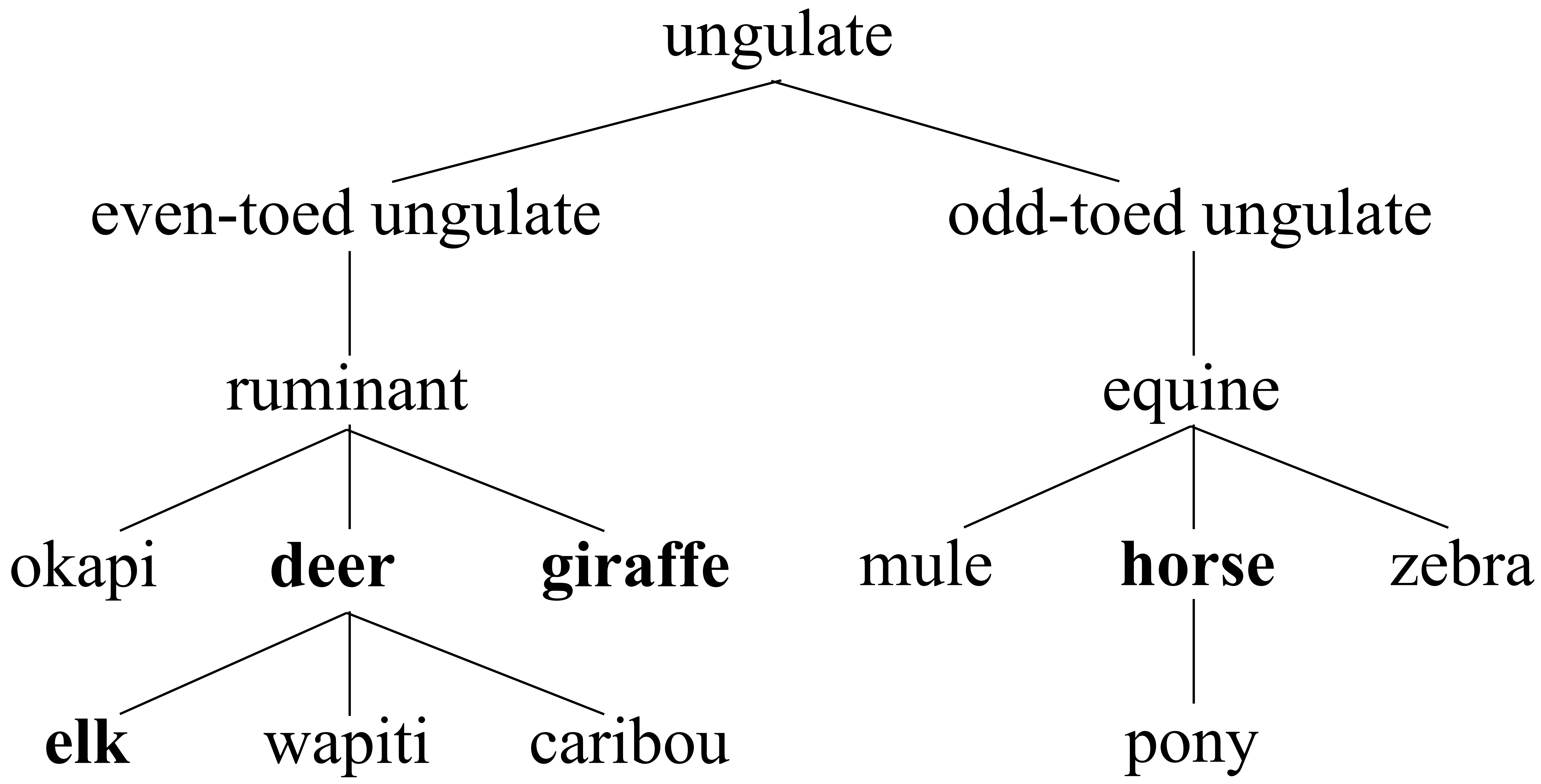
WordNet

- Semantic dictionary of (mostly) English words, interlinked by semantic relations
- Includes rich linguistic information
  - part of speech, word senses, synonyms, hypernyms/ hyponyms, meronyms, distributional related forms, …
- Machine-readable, freely available

Semantic Similarity Using WordNet

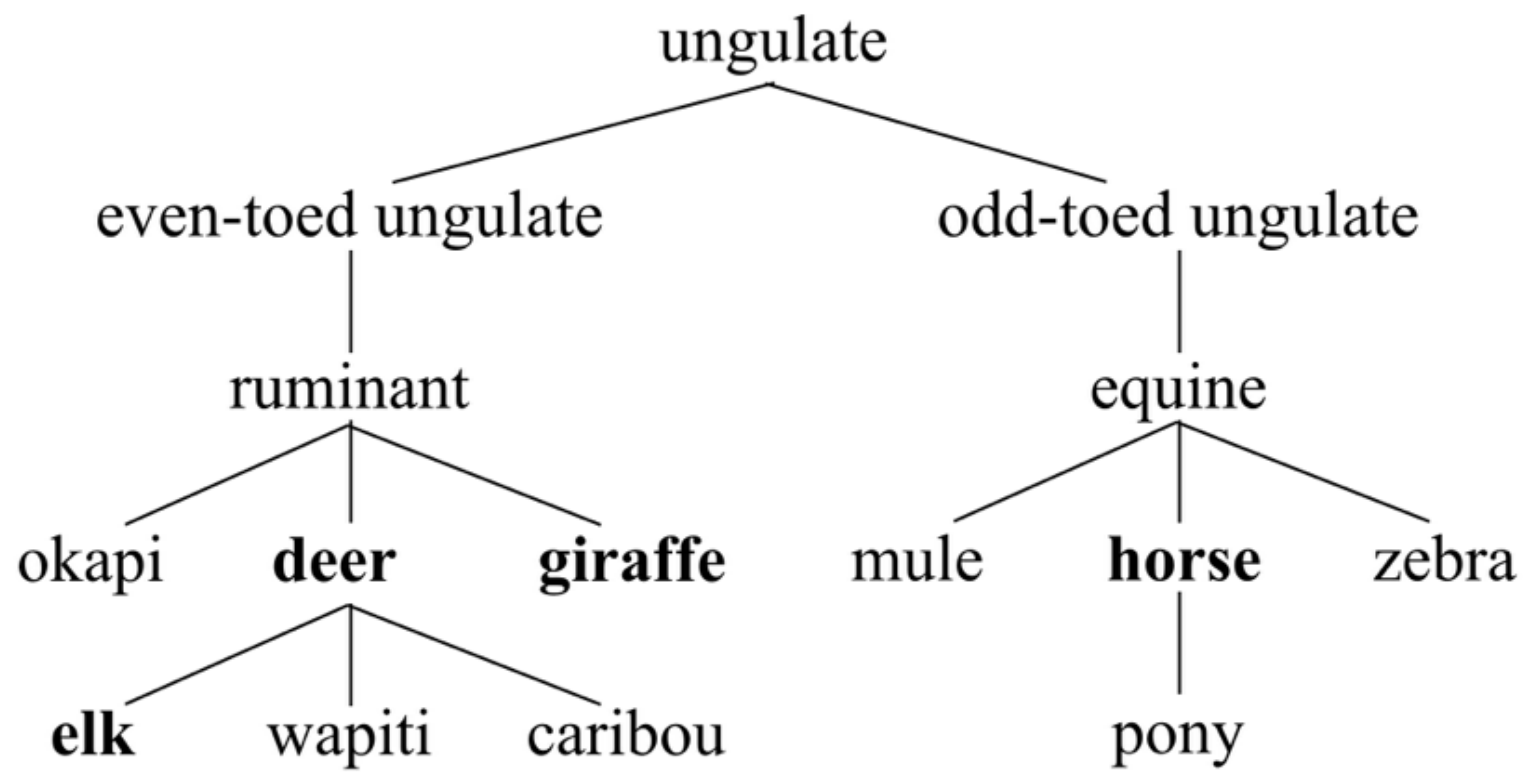
- WordNet organizes information in a hierarchy
- Many similarity measures use the hierarchy in some way
- Verbs, nouns, adjectives all have separate hierarchies

Coming back to our deer example



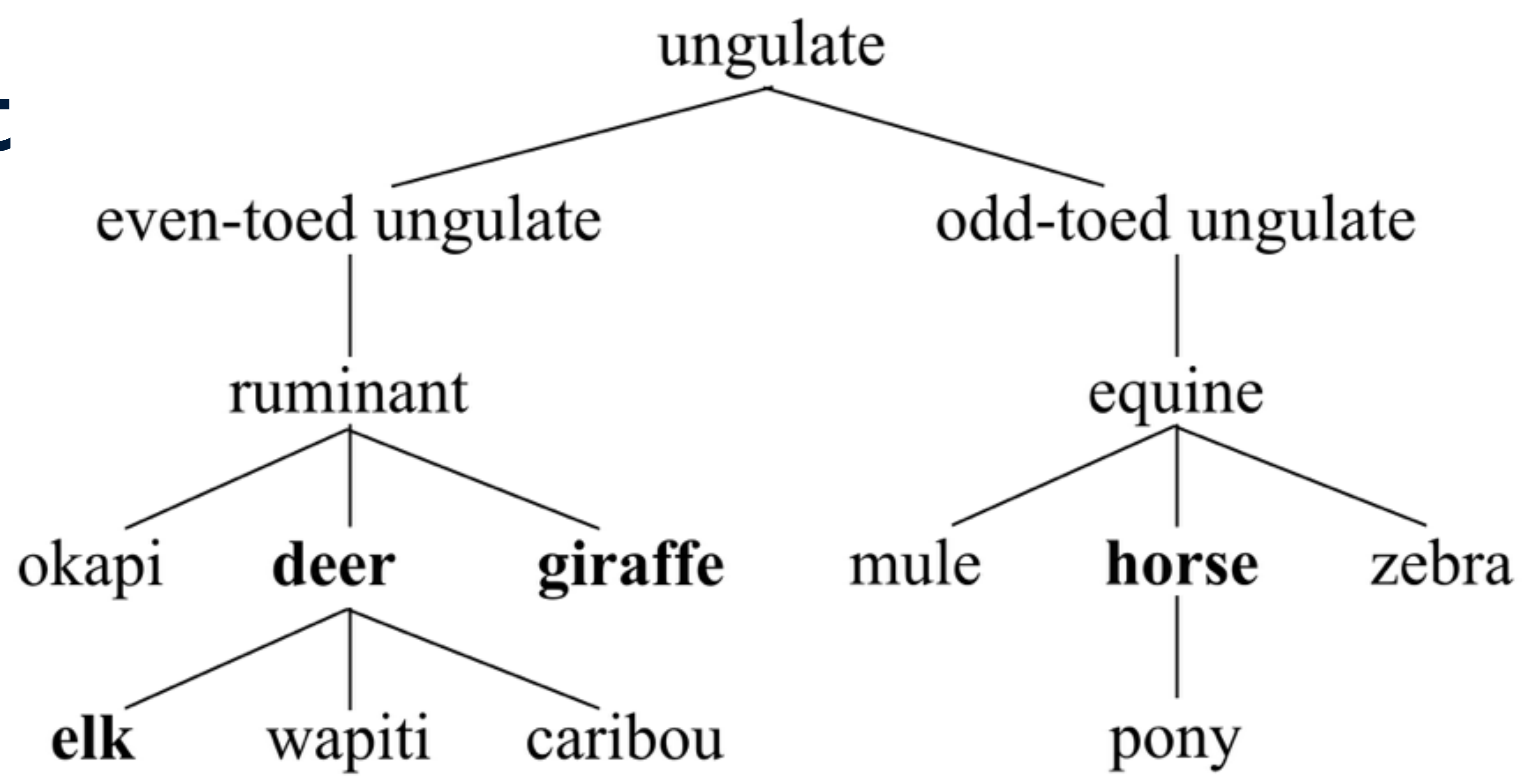
Path Similarity

- 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



Lowest Common Subsumer (LCS)

- Find the closest ancestor to both concepts
  - LCS(deer, elk) = deer
  - LCS(deer, giraffe) = ruminant
  - LCS(deer, horse) = ungulate



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
```

Collocations and Distributional Similarity

- "You know a word by the company it keeps" [Firth, 1957]
- Two words that frequently appears in similar contexts are more likely to be semantically related
  - The friends met at a café.
  - Shyam met Ray at a pizzeria.
  - Let's meet up near the coffee shop.
  - The secret meeting at the restaurant soon became public.

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, …

Strength of association between words

- How frequent are these?
  - Not similar if two words don’t occur together often
- Also important to see how frequent are individual words
  - ‘the’ is very frequent, so high chances it co-occurs often with every other word
- Pointwise Mutual Information $\mathrm{PMI}(w,c) = \log\, [P(w,c) \,/\, P(w)P(c)]$

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