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

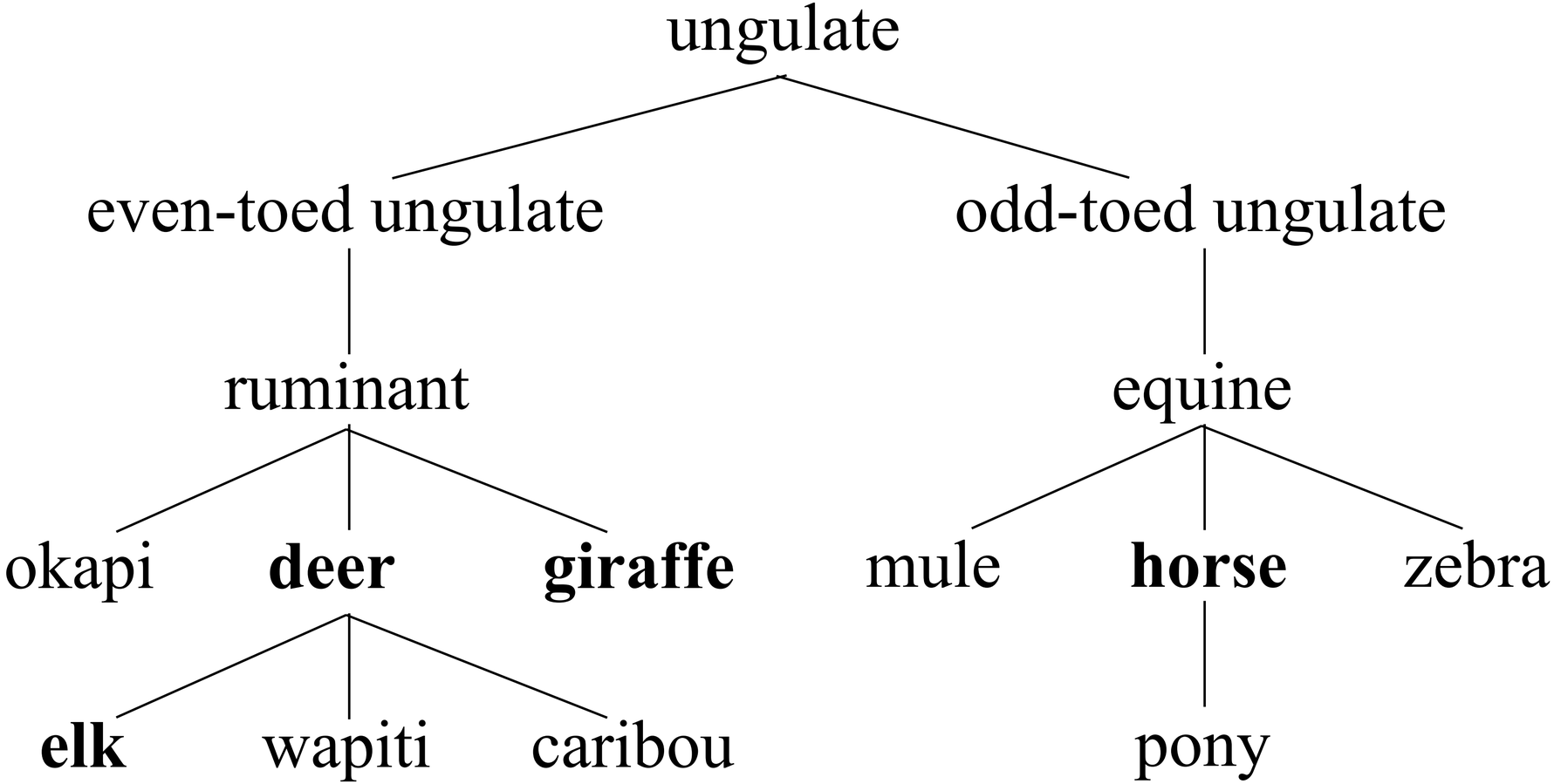
WordNet

- 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

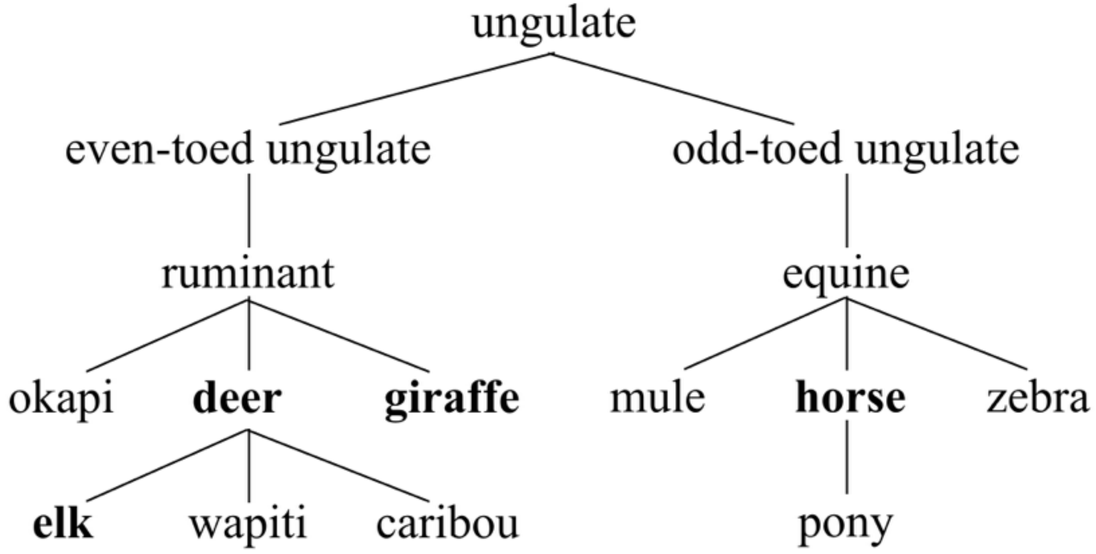
- WordNet organizes information in a hierarchy
- Many similarity measures use the hierarchy in some way
- Verbs, nouns, adjectives all have separate hierarchies

Coming back to our deer example



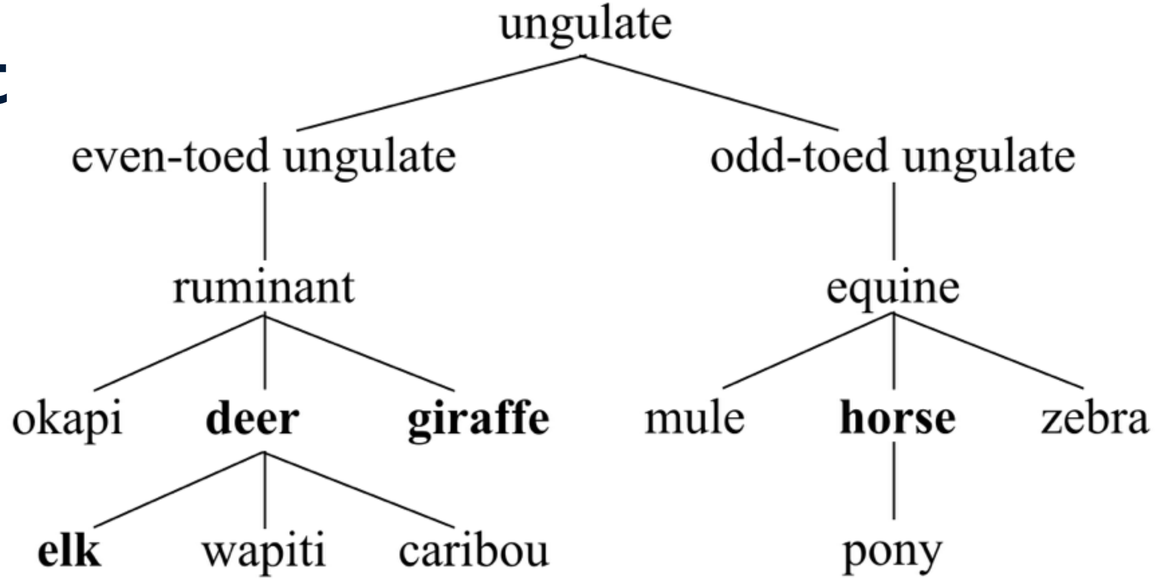
Path Similarity

- 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 $PMI(w,c) = \log [P(w,c) / P(w)P(c)]$

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

Applied Text Mining in Python

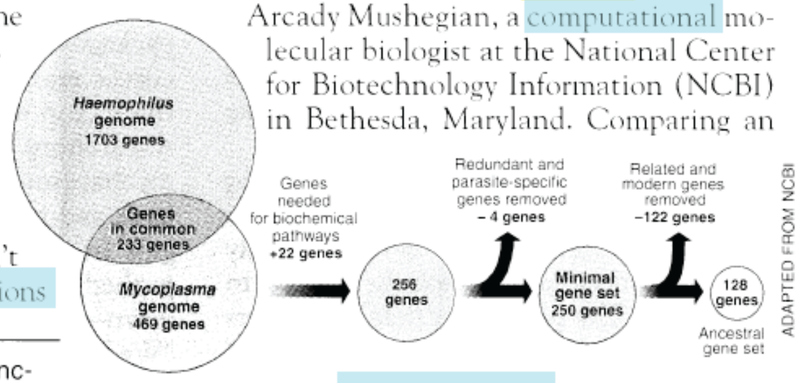
Topic modeling

Documents Exhibit Multiple Topics

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

(Figure courtesy Prof. David Blei)

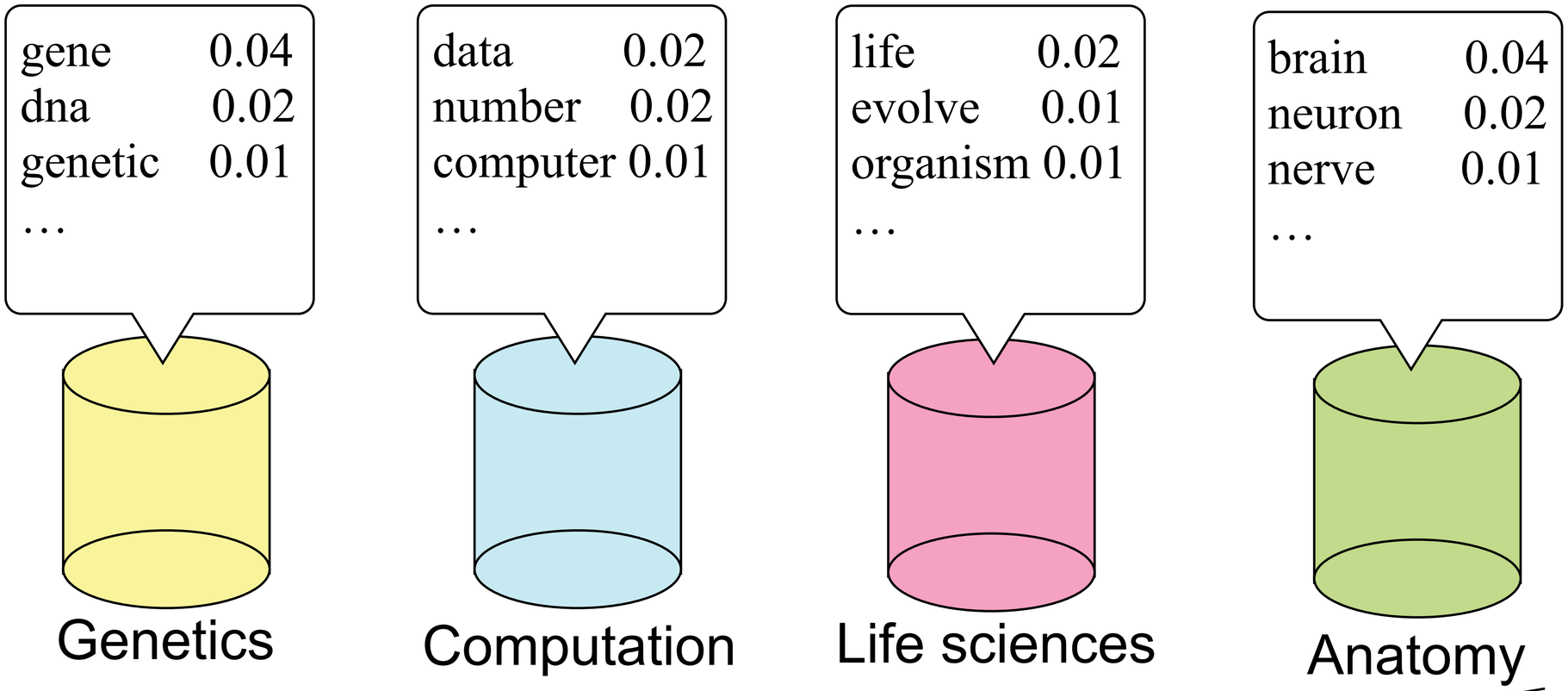
Latent Dirichlet Allocation
(Blei et al., '03)

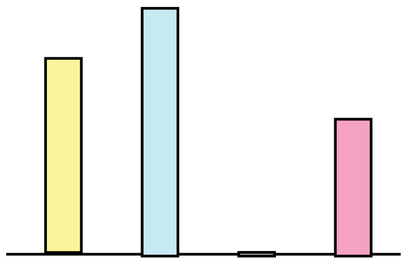
Topic 1: Genetics
gene, sequence, genome, …

Topic 2: Computation
number, computer, analysis, …

Topic 3: Life sciences
life, survive, organism, …

Intuition: Documents as a mixture of topics





What is Topic Modeling?

- A course-level analysis of what's in a text collection
- Topic : the subject (theme) of a discourse
- Topics are represented as a word distribution
- A document is assumed to be a mixture of topics

More examples of topics

human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

(Figure courtesy Prof. David Blei)

What is Topic Modeling? (2)

- What's known:
  - The text collection or corpus
  - Number of topics
- What's not known:
  - The actual topics
  - Topic distribution for each document

What is Topic Modeling? (3)

- Essentially, text clustering problem
  - Documents and words clustered simultaneously

- Different topic modeling approaches available
  - Probabilistic Latent Semantic Analysis (PLSA) [Hoffman '99]
  - Latent Dirichlet Allocation (LDA) [Blei, Ng, and Jordan, '03]

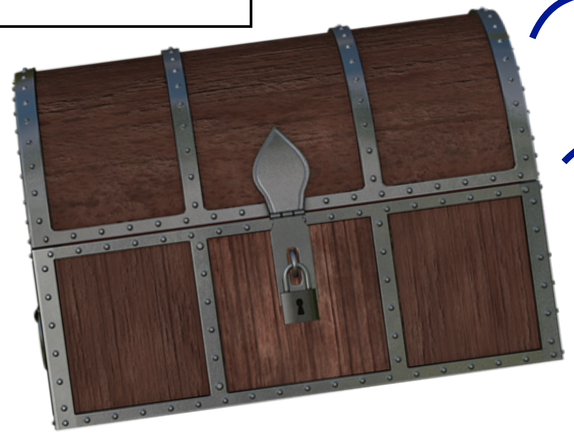
Applied Text Mining in Python

Generative models and LDA

Generative Models for Text

Pr(text | model)

the	0.1
is	0.07
harry	0.05
potter	0.04
movie	0.04
plot	0.02
time	0.01
rowling	0.01

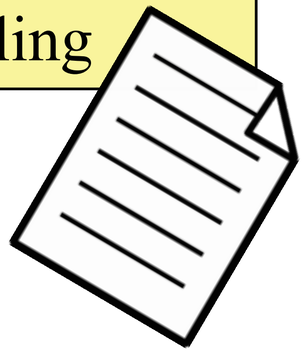


Inference, Estimation

the harry potter movie is harry is

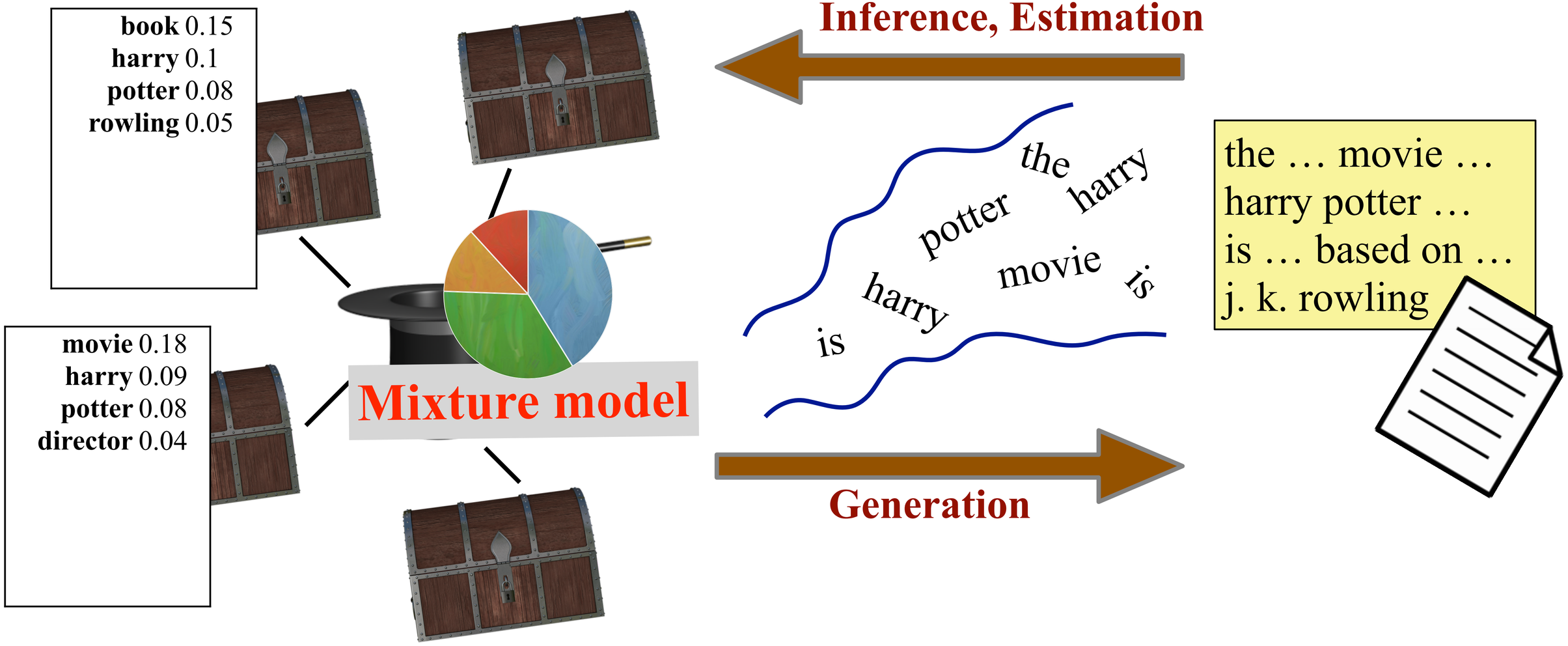
Generation

the … movie …
harry potter …
is … based on …
j. k. rowling



Generative Models can be Complex

Pr(text | model)



Latent Dirichlet Allocation (LDA)

- Generative model for a document d
  - Choose length of document d
  - Choose a mixture of topics for document d
  - Use a topic's multinomial distribution to output words to fill that topic's quota

Topic Modeling in Practice

- How many topics?
  - Finding or even guessing the number of topics is hard
- Interpreting topics
  - Topics are just word distributions
  - Making sense of words / generating labels is subjective

Topic Modeling: Summary

- Great tool for exploratory text analysis
  - What are the documents (tweets, reviews, news articles) about?
- Many tools available to do it effortlessly in Python

Working with LDA in Python

- Many packages available, such as gensim, lda
- Pre-processing text
  - Tokenize, normalize (lowercase)
  - Stop word removal
  - Stemming
- Convert tokenized documents to a document - term matrix
- Build LDA models on the doc-term matrix

Working with LDA in Python (2)

- **doc_set: set of pre-processed text documents**

```
import gensim
from gensim import corpora, models

dictionary = corpora.Dictionary(doc_set)
corpus = [dictionary.doc2bow(doc) for doc in doc_set]
ldamodel = gensim.models.ldamodel.LdaModel (corpus, num_topics=4,
id2word=dictionary, passes=50)
print(ldamodel.print_topics(num_topics=4, num_words=5))
```

- **ldamodel can also be used to find topic distribution of documents**

```
topic_dis = ldamodel[new_doc]
```

Take Home Concepts

- Topic modeling is an exploratory tool frequently used for text mining
- Latent Dirichlet Allocation is a generative model used extensively for modeling large text corpora
- LDA can also be used as a feature selection technique for text classification and other tasks

Applied Text Mining in Python

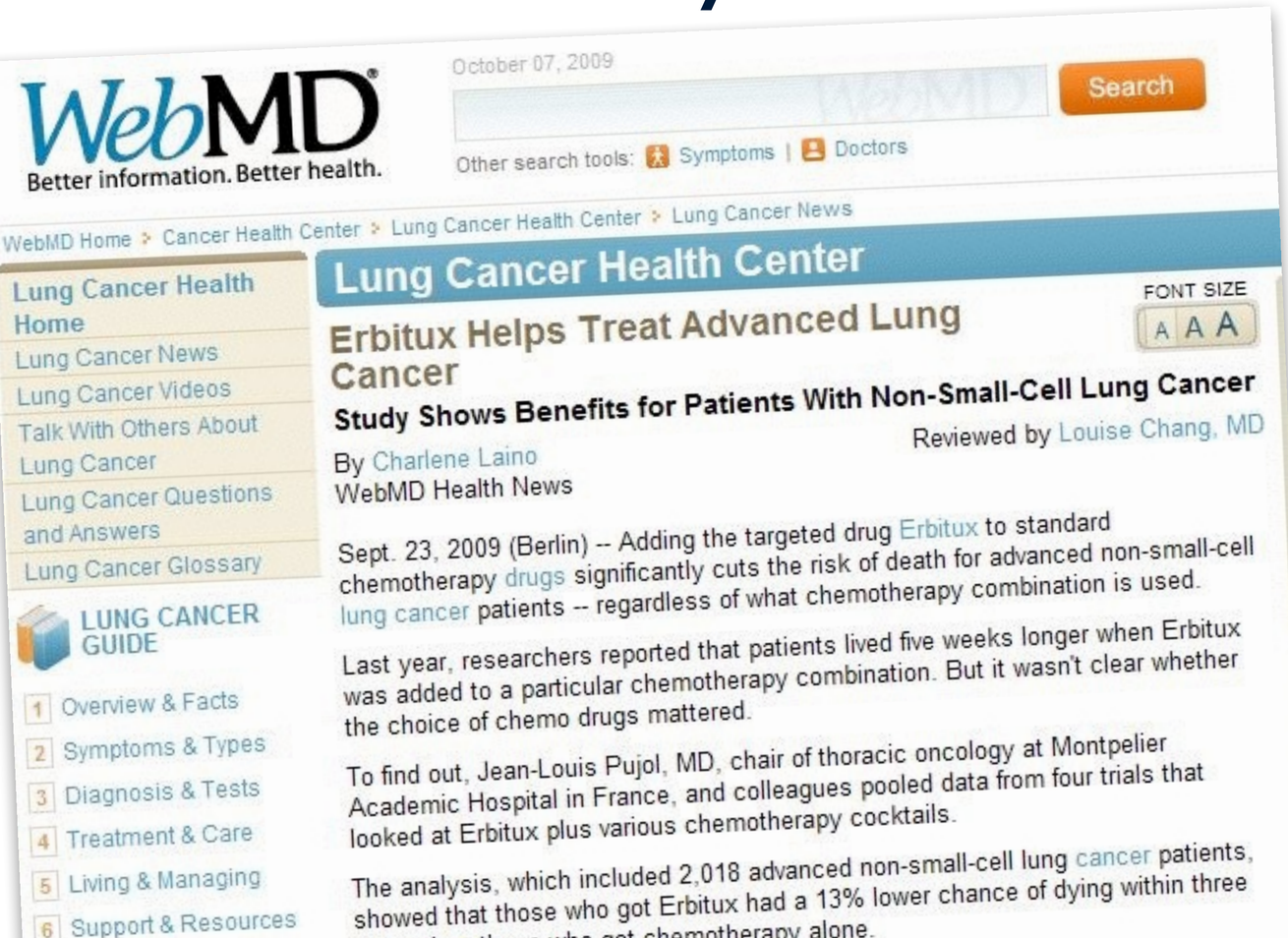
Information Extraction

Information is hidden in free-text

- Most traditional transactional information is structured
- Abundance of unstructured, freeform text
- How to convert unstructured text to structured form?

Information Extraction

- **Goal: Identify and extract fields of interest from free text**



Erbitux helps treat lung cancer

Author: Charlene Laino

Reviewer: Louise Chang, MD

Sept. 23, 2009 | Berlin | …

Fields of Interest

- Named entities
  - [NEWS] People, Places, Dates, …
  - [FINANCE] Money, Companies, …
  - [MEDICINE] Diseases, Drugs, Procedures, …
- Relations
  - What happened to who, when, where, …

Named Entity Recognition

- **Named entities**: Noun phrases that are of specific type and refer to specific individuals, places, organizations, …
- **Named Entity Recognition**: Technique(s) to identify all mentions of pre-defined named entities in text
  - Identify the mention / phrase: *Boundary detection*
  - Identify the type: *Tagging / classification*

Examples of Named Entity Recognition Tasks

The patient is a 63-year-old female with a three-year history of bilateral hand numbness and occasional weakness.

Within the past year, these symptoms have progressively gotten worse, to encompass also her feet.

She had a workup by her neurologist and an MRI revealed a C5-6 disc herniation with cord compression and a T2 signal change at that level.

Approaches to identify named entities

- Depends on kinds of entities that need to be identified

- For well-formatted fields like date, phone numbers: Regular expressions (Recall Week 1)
- For other fields: Typically a machine learning approach

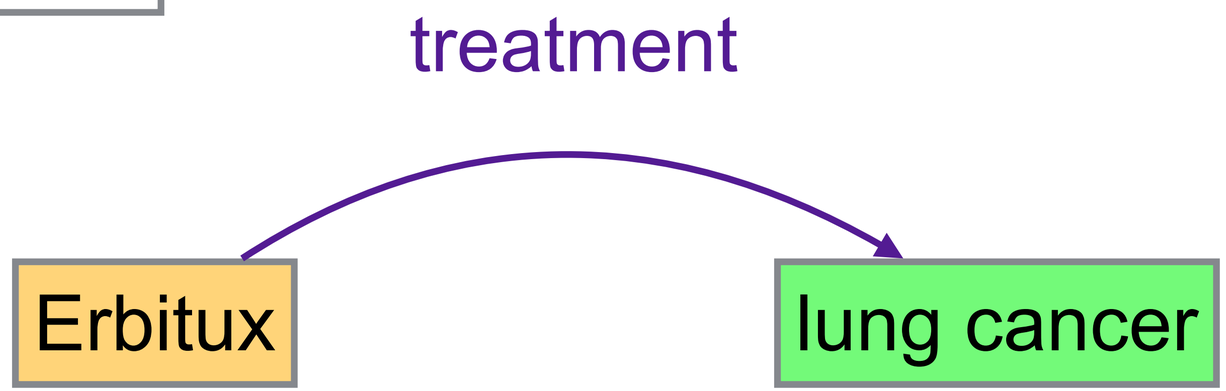
Person, Organization, Location/GPE

- Standard NER task in NLP research community
- Typically a four-class model
  - PER
  - ORG
  - LOC / GPE
  - Other / Outside (any other class)

Relation Extraction

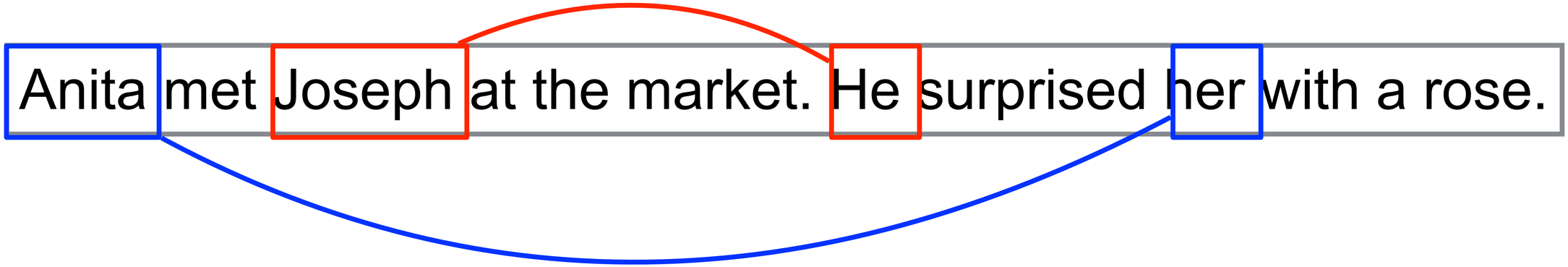
- **Identify relationships between named entities**

Erbitux helps treat lung cancer



Co-reference Resolution

- **Disambiguate mentions and group mentions together**



Question Answering

- Given a question, find the most appropriate answer from the text
  - What does Erbitux treat?
  - Who gave Anita the rose?
- Builds on named entity recognition, relation extraction, and co-reference resolution

Take Home Concepts

- Information Extraction is important for natural language understanding and making sense of textual data
- Named Entity Recognition is a key building block to address many advanced NLP tasks
- Named Entity Recognition systems extensively deploy supervised machine learning and text mining techniques discussed in this course