# Statistical methods for NLP
# Unsupervised and semisupervised methods



Richard Johansson

March 5, 2015

# information

- the machine translation lecture has been moved to **March 12**
- L308, usual time

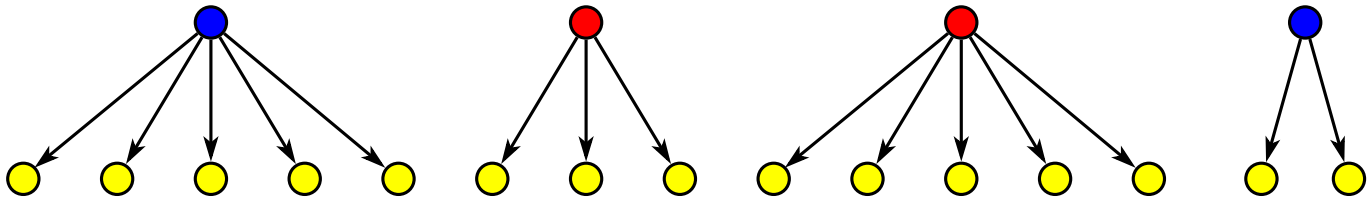
# overview of today's lecture

- what to do when we have little or no labeled data
- or in general: when some part of our model is **unobserved**

# overview

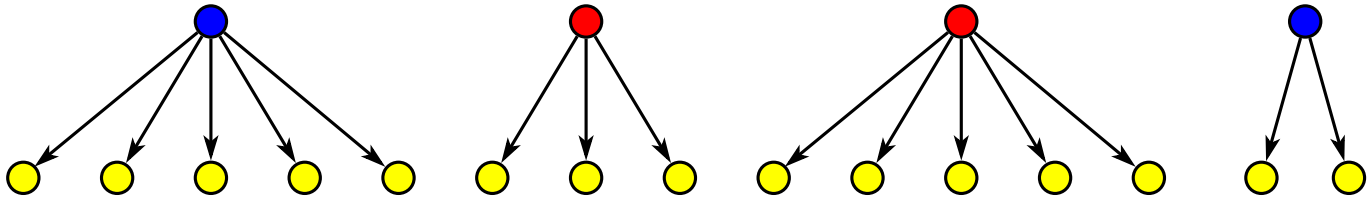
# what kind of information is available?

- **supervised** learning: the labels are given

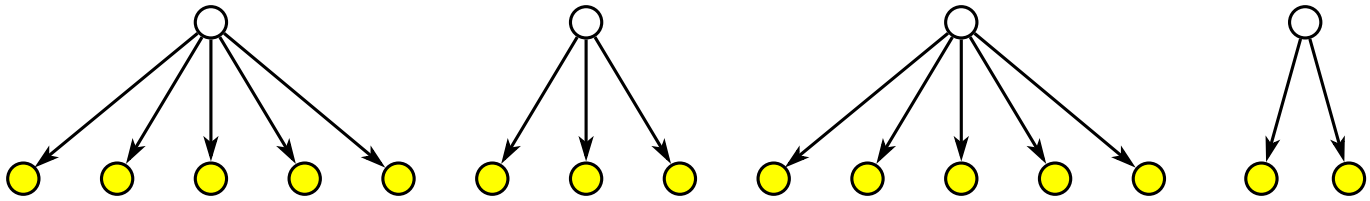


# what kind of information is available?

- **supervised** learning: the labels are given

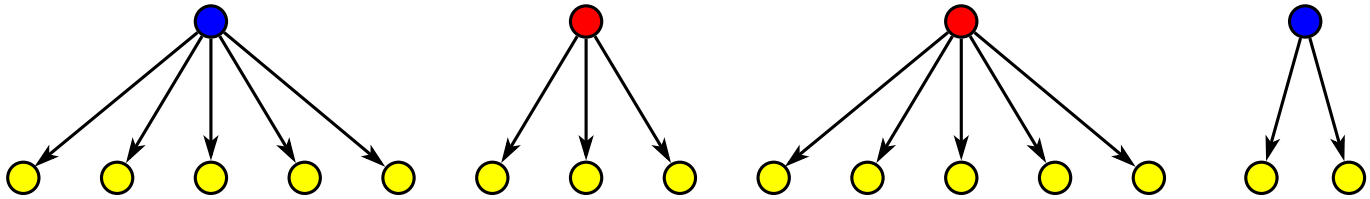


- **unsupervised** learning (**clustering**): the labels are not given

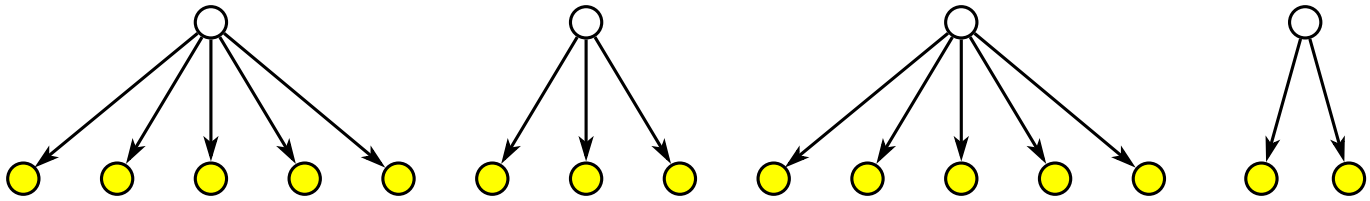


# what kind of information is available?

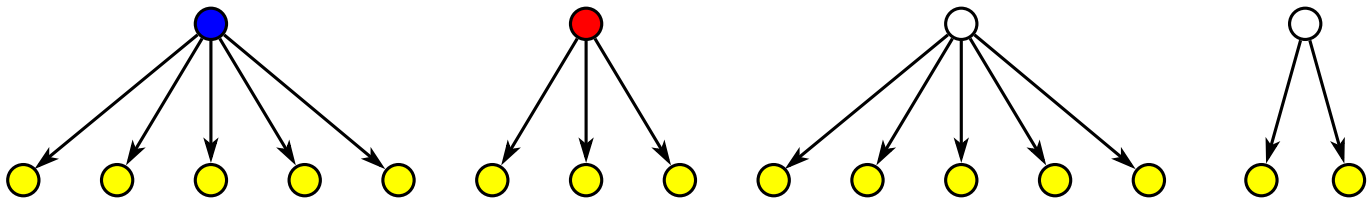
- **supervised** learning: the labels are given



- **unsupervised** learning (**clustering**): the labels are not given

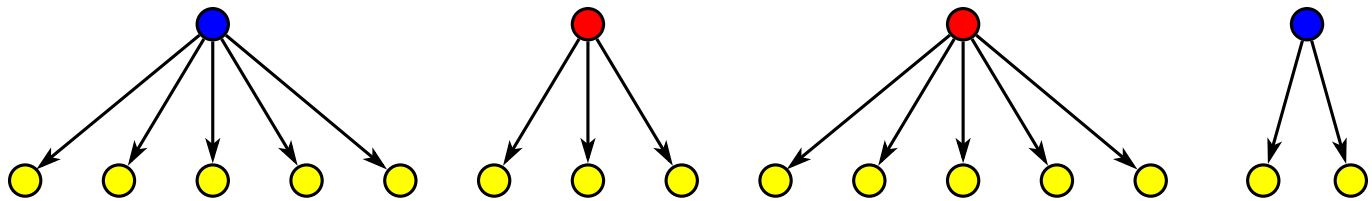


- **semisupervised** learning: some of the labels are given



# overview

# estimation in Naive Bayes, revisited



- Naive Bayes:

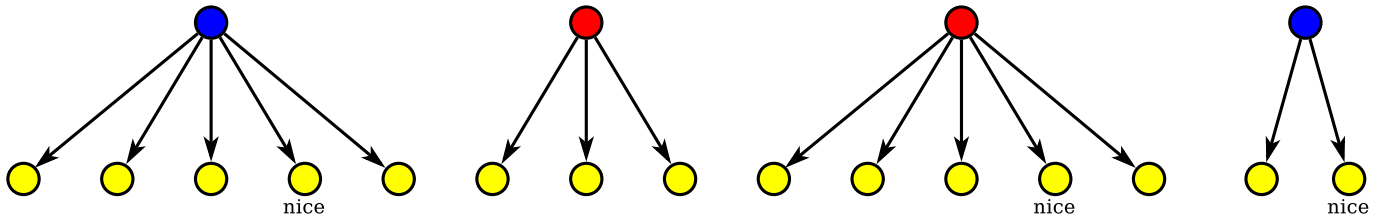
$$P(\text{document}, label) =$$

$$P(f_1, \ldots, f_n, label) = P(label) \cdot P(f_1, \ldots, f_n | label)$$

$$= P(label) \cdot P(f_1 | label) \cdot \ldots \cdot P(f_n | label)$$

- how do we estimate the probabilities?
  - **maximum likelihood**: set the probabilities so that the probability of the data is maximized

# estimation in Naive Bayes: supervised case



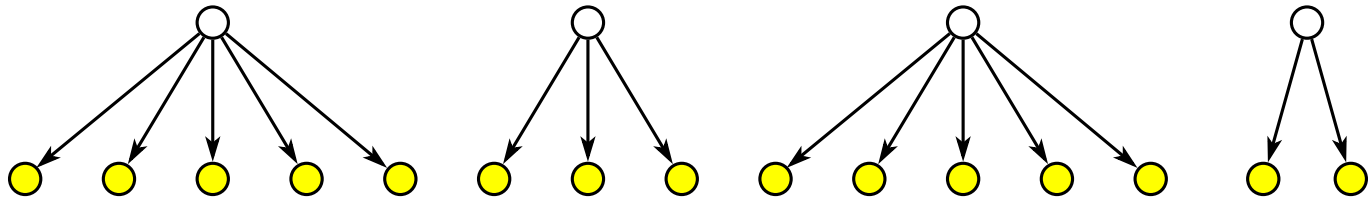

- how do we estimate $P(\text{positive})$?

$$P_{\text{MLE}}(\text{positive}) = \frac{\text{count}(\text{positive})}{\text{count}(\text{all})} = \frac{2}{4}$$

- how do we estimate $P(\text{“nice”}|\text{positive})$?

$$P_{\text{MLE}}(\text{“nice”}|\text{positive}) = \frac{\text{count}(\text{“nice”}, \text{positive})}{\text{count}(\textit{any word}, \text{positive})} = \frac{2}{7}$$

# what if we are missing labeled data?



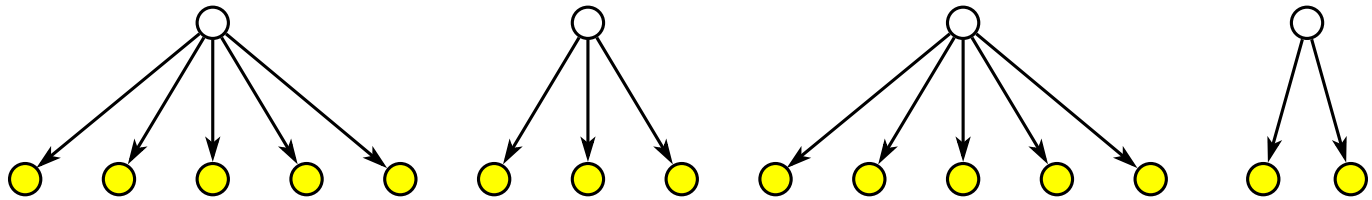
- we stay in the maximum likelihood framework
- in the supervised case, we maximize

$$P(\text{doc}_1, \text{lbl}_1) \cdots P(\text{doc}_n, \text{lbl}_n)$$

- in the unsupervised case, we do the same, only that we don't observe the document labels, so we instead maximize

$$P(\text{doc}_1) \cdots P(\text{doc}_n)$$

# maximizing the likelihood in the unsupervised case



- what is the probability of a document?
  - sum over all possible labels
  - e.g. with sentiment labels:

$$P(\text{doc}) = P(\text{doc}, \text{POS}) + P(\text{doc}, \text{NEG})$$

- ...so the likelihood is:

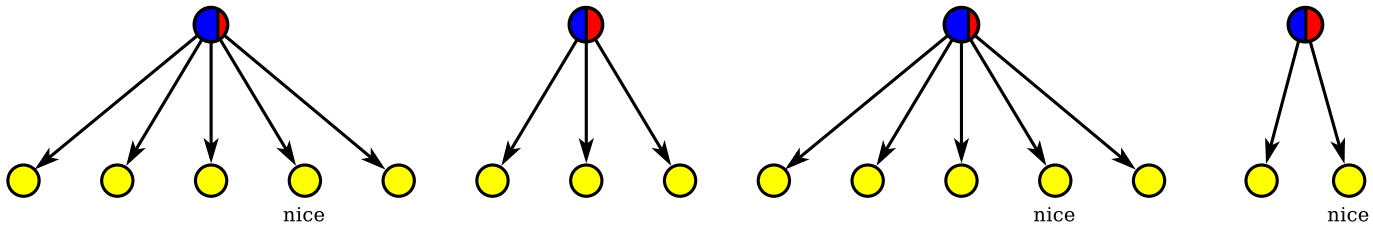
$$\big(P(\text{doc}_1, \text{POS}) + P(\text{doc}_1, \text{NEG})\big) \cdots \big(P(\text{doc}_n, \text{POS}) + P(\text{doc}_n, \text{NEG})\big)$$

- this formula is more complex and if we want to maximize it, there is no nice and clean solution as in the supervised case

# the Expectation–Maximization algorithm

- **Expectation–Maximization** is an algorithm that tries to maximize likelihood when there are unobserved variables
- it is a "circular" two-step algorithm:
  - **Expectation**: using our current estimates, compute label probabilities for each documents and use them as "soft counts"
    - for instance, if a document has a probability of 40% of being positive, then we count it as 40% of a positive document
  - **Maximization**: using the soft counts, compute probability estimates with the normal procedure
- a chicken-and-egg problem
  - . . . so we need to initialize either the soft counts or the probabilities, and then start at E or M

# soft counts example, Naive Bayes



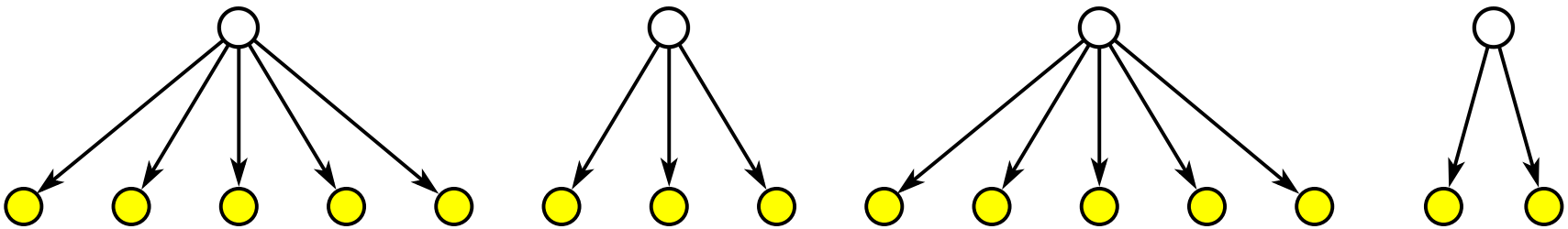

- let's assume that our classifier assigned the probabilities 0.8, 0.5, 0.7, and 0.6, respectively, for the documents belonging to the positive class
- then:

$$P_{\text{MLE}}(\text{positive}) = \frac{\text{count}(\text{positive})}{\text{count}(\text{all})} = \frac{0.8 + 0.5 + 0.7 + 0.6}{4}$$

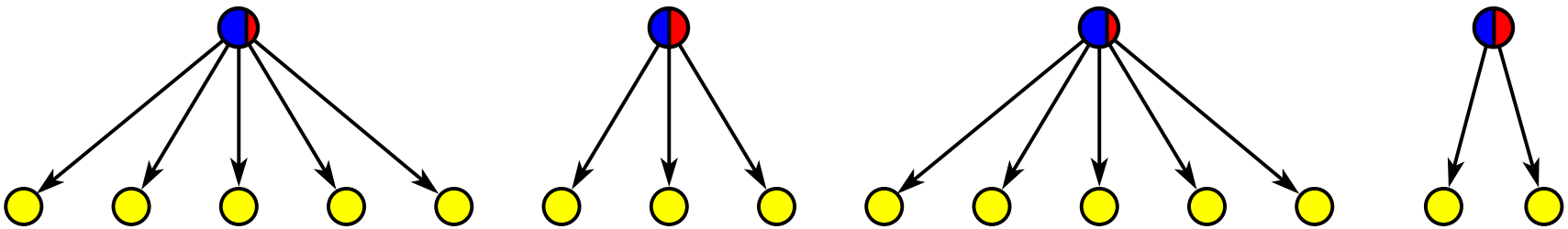
$$P_{\text{MLE}}(\text{“nice”}|\text{positive}) = \frac{\text{count}(\text{“nice”}, \text{positive})}{\text{count}(\textit{any word}, \text{positive})}$$

$$= \frac{0.8 + 0.7 + 0.6}{0.8 \cdot 5 + \ldots + 0.6 \cdot 2}$$

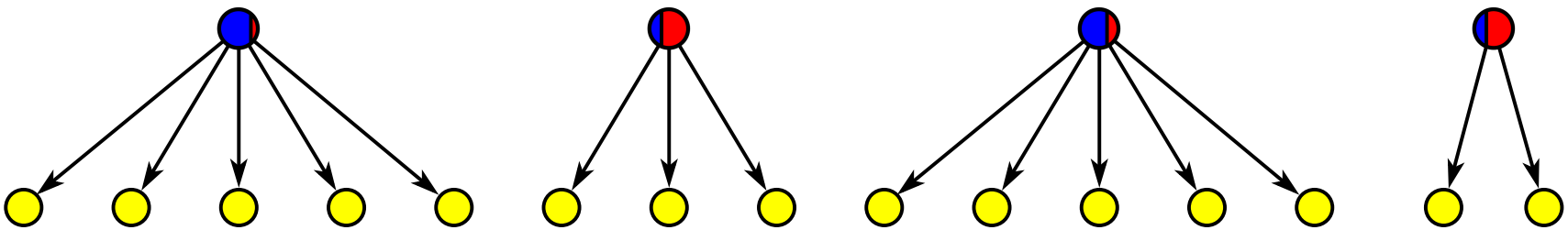
# EM iteration for Naive Bayes, example



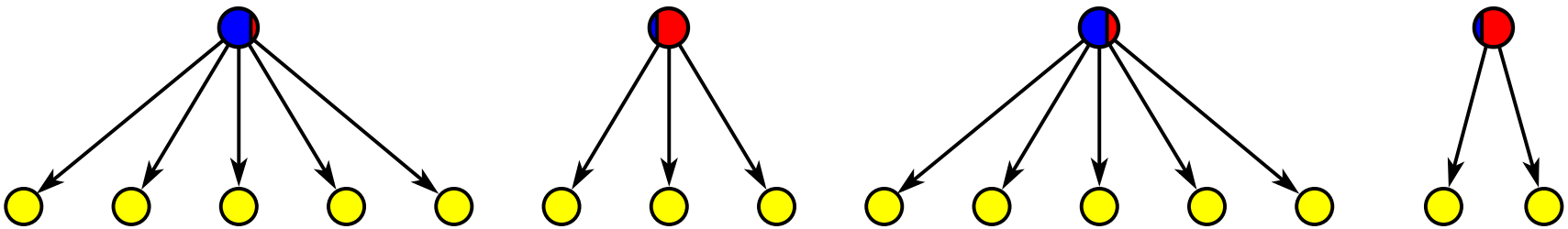
# EM iteration for Naive Bayes, example



# EM iteration for Naive Bayes, example

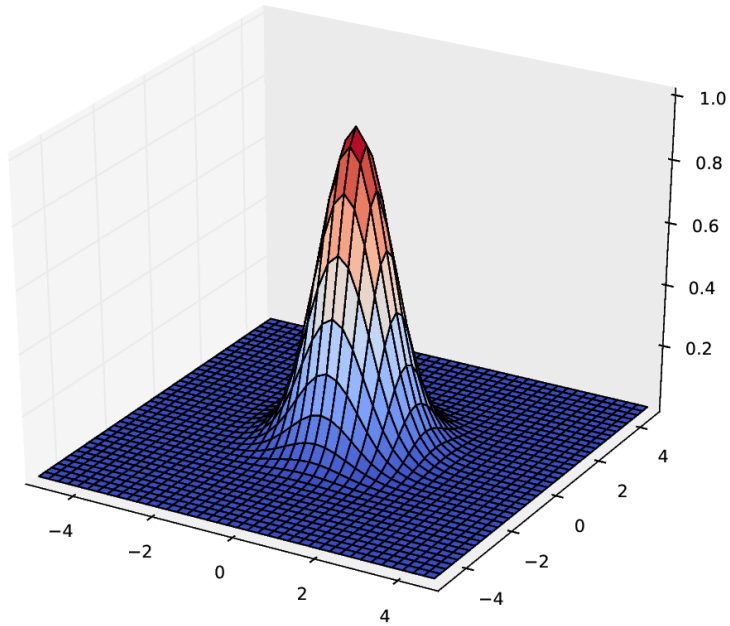


# EM iteration for Naive Bayes, example



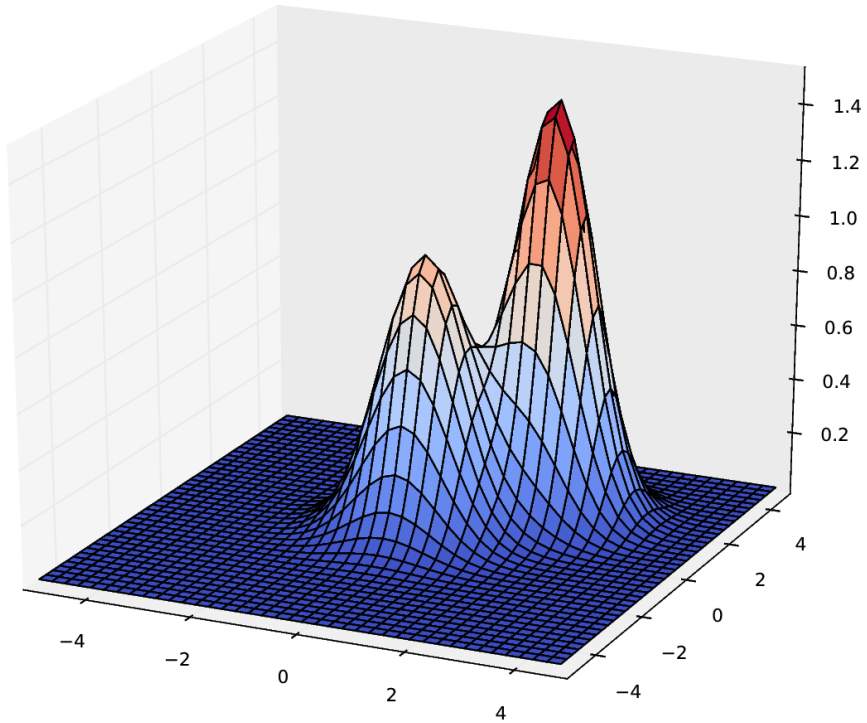
# nice property of EM

- **theorem**: each time we carry out the E and M steps, the likelihood won't decrease
- EM will converge (stop) at some point
  - we can climb "uphill" until we reach the top



# not-so-nice property of EM

- . . . but the likelihood may end up in a **local maximum** rather than the true maximum
  - there might be tops that are higher




- initialization will determine where we end up!

# EM initialization

- EM may give completely different results depending on how we initialize
- . . . which will depend on our situation
  - unsupervised setting: typically we will initialize randomly, so it may be good to run the algorithm several times
  - semi-supervised setting: typically we will initialize by using the labeled data only
- we can use a knowledge source if available
- in some cases (e.g. machine translation, next lecture) it can be useful to use a simple model for initializing the estimation of a complex model

# semi-supervised experiments

- experiments using the dataset from the assignments
  - positive vs. negative book reviews (10/1500): 0.598 $\rightarrow$ 0.643
  - DVD reviews vs. music reviews (10/3500): 0.648 $\rightarrow$ 0.913
  - positive vs. negative reviews (100/1500): **0.625 $\rightarrow$ 0.452**
- EM does not always improve the result!
  - depends on properties of the data and the difficulty of the task
  - in the bad example above, EM seems to pick up review types rather than sentiments

# unsupervised experiment

- DVD reviews vs. music reviews
- accuracy (assuming category A is DVDs): 0.938

category A:

```
episodes 3.639010348
novel 3.61869012429
sequel 3.57220166914
seagal 3.33618534235
suspense 3.23906073358
vampire 3.19579018682
buffy 3.17306201864
plot 3.16664944584
thriller 3.14954723164
premise 3.12519741486
```

category B:

```
vocals -4.15311759228
albums -4.12576126936
album -3.98345604473
catchy -3.98199815327
acoustic -3.86295373766
punk -3.80731612078
guitars -3.79254346553
lyrics -3.65137587064
remix -3.64054296621
lp -3.6211479113
```

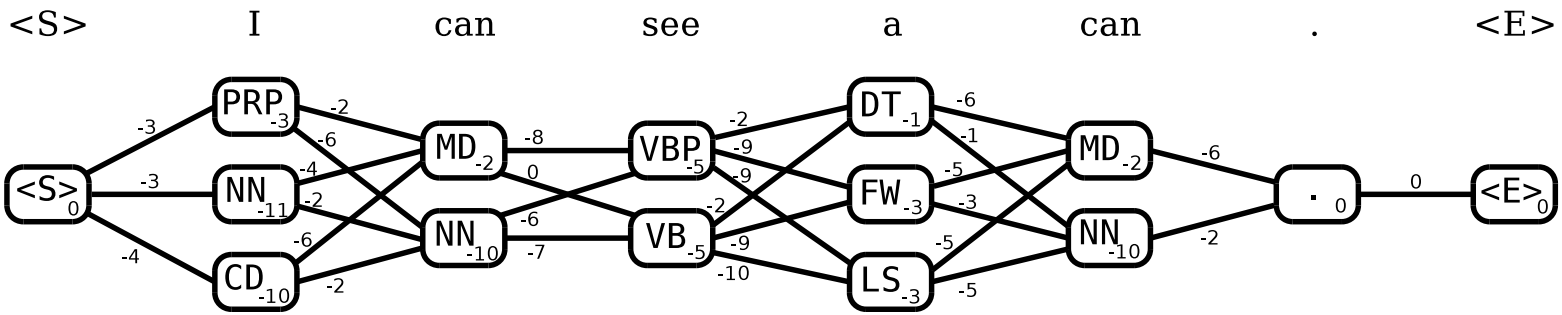
# are the results meaningful?

- if we cluster review documents: will we cluster by sentiment or by topic? Or by gender of author?
- in algorithms such as $k$-means and Naive Bayes+EM: the results depend a lot on initialization, and the number of classes
- also other tricks such as feature weighting and filtering: stop word removal, TF-IDF, ...
- a hard problem in unsupervised learning in general: **evaluation**

# EM in the general case

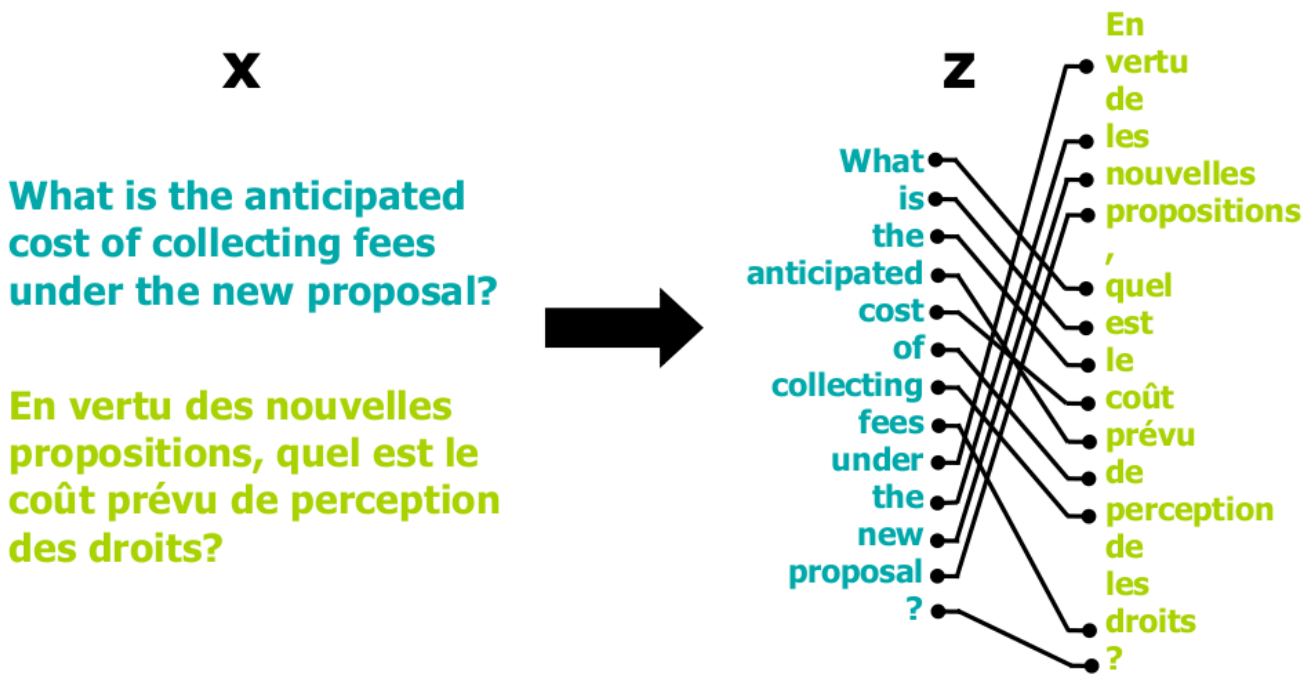
- EM is a general recipe that can be applied to a wide range of problems, not just classification with Naive Bayes
- always the same form:
  - E: compute soft counts
  - M: estimate probabilities
- . . . but exactly how the steps are carried out depends on the problem
- in particular, special tricks might be needed to get the soft counts:
  - part-of-speech tagging: the **forward–backward** (a.k.a. **Baum–Welch**)
  - PCFG parsing: the **inside–outside** algorithm
  - these algorithms are similar to Viterbi and CKY, respectively

# tagging example



# next lecture and VG assignment 2

- word alignment for machine translation
- the observed part: sentence pairs
- the unseen part: word-to-word alignments



# hard EM

- EM is based on "soft counts" as we saw before
- what if we cheat and just find the maximal probability?
- then we have a variant known as **hard EM** or **self-training**
  - **Expectation**: using our current estimates, compute label probabilities for each document and find the labels with the maximal probability
    - for instance, if a document has a probability of 60% of being positive and 40% of being negative, then we count it as positive
  - **Maximization**: using the guesses, compute probability estimates with the normal procedure
- **note**: this doesn't really require a probabilistic model

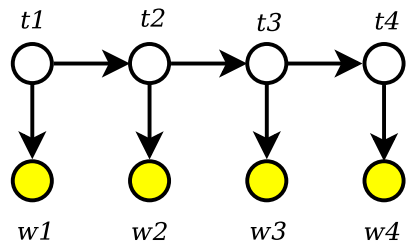
# overview

# $k$-means

- (see other slides)

# Brown clustering

- the **Brown** algorithm is a clustering method for **words**:
  - start by putting each word into a cluster
  - merge clusters to increase HMM language model probability of our corpus




- there is a popular implementation by Percy Liang:
  - `http://cs.stanford.edu/~pliang/software/` – under "Word clustering"

Brown et al. *Class-Based n-gram Models of Natural Language*. Computational Linguistics, 1992.

# example: clusters from book reviews

- I created a corpus of 865,000 Amazon book reviews and ran Liang's software to create 2,000 word clusters
- . . . after a few days, it finished
- here are a couple of examples of clusters

cluster 1000010010111:

| | |
|---|---|
| time-consuming | 225 |
| repugnant | 234 |
| unnerving | 240 |
| objectionable | 243 |
| anticlimactic | 244 |
| reprehensible | 258 |
| anti-climactic | 270 |
| deceiving | 289 |
| disrespectful | 299 |
| dissapointing | 308 |

cluster 1011111100011:

| | |
|---|---|
| maine | 1758 |
| turkey | 1796 |
| manhattan | 1860 |
| boston | 3704 |
| florida | 3764 |
| chicago | 4535 |
| london | 8383 |
| paris | 6329 |
| heaven | 5864 |
| california | 7094 |

# Brown clusters in classifiers

- in NLP, we very often use words as features
  - document classification
  - parsing and POS tagging
  - named entity extraction
- problems:
  - there are a lot of possible words
  - most of them occur very rarely
- Brown clusters can help us generalize:

```
1100111011 Gothenburg
1100111011 Ashgabat
110011100 Sydney
110011100 Paris
```

J. Turian, L. Ratinov, Y. Bengio. *Word representations: A simple and general method for semi-supervised learning.* ACL 2010.

# experiments: book reviews

- I trained a classifier as in Assignment 1 on the book review subset of the corpus:
  - only the words: 0.785
  - clusters IDs instead of words: 0.797
  - words and cluster IDs: 0.801
- examples of some useful clusters:
  - excellent, excellant, excelent, inetersting, ...
  - marvelous, wonderful, marvellous, one-of-a-kind, ...
  - stupid, silly, ridiculous, useless, dumb, ...
  - annoying, confusing, disappointing, frustrating, ...

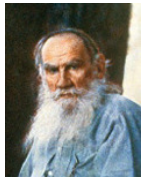
# overview

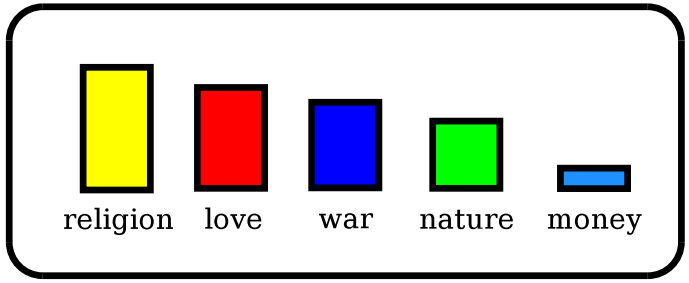
# topic modeling

- for documents, clustering is sometimes too simple
- a document contains more than one "topic"
  - example: a camera review has a camera topic and a sentiment topic
- **Topic modeling** for a corpus of documents:
  1. find the "topics" in the corpus
     - a topic is a probability distribution over words
  2. analyze documents as composed by the topics
- the most popular topic model is called Latent Dirichlet Allocation (**LDA**)
  - it is a multilayered generative model that is a bit more complex than e.g. Naive Bayes

# generative story in LDA

- Tolstoy's preferred topics:

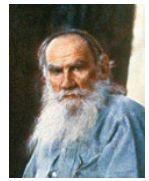





| religion | | love | | war | | nature | | money | |
|---|---|---|---|---|---|---|---|---|---|
| God | 0.06 | love | 0.05 | war | 0.05 | forest | 0.07 | money | 0.05 |
| faith | 0.04 | marry | 0.04 | soldier | 0.04 | flower | 0.05 | ruble | 0.04 |
| believe | 0.03 | wedding | 0.04 | battle | 0.03 | grass | 0.04 | gold | 0.04 |
| sacrifice | 0.02 | heart | 0.03 | gun | 0.03 | open | 0.04 | pay | 0.03 |
| pray | 0.02 | miss | 0.03 | brave | 0.03 | lake | 0.03 | debt | 0.03 |
| priest | 0.02 | dear | 0.02 | wound | 0.03 | tundra | 0.03 | earn | 0.02 |
| inner | 0.01 | propose | 0.02 | resist | 0.02 | tree | 0.02 | coin | 0.02 |
| devotion | 0.01 | kiss | 0.01 | kill | 0.02 | sprin | 0.01 | contract | 0.01 |
| ... | | ... | | ... | | ... | | ... | |

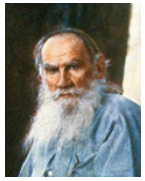
# generative story in LDA

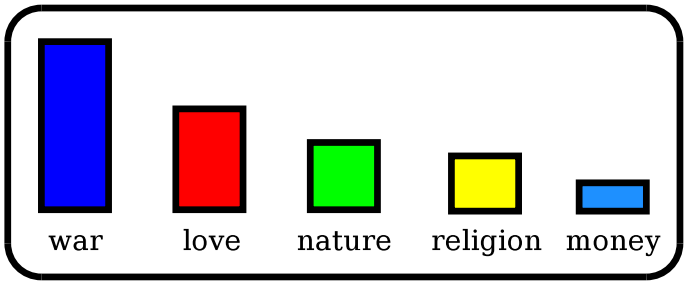
- when writing *War and Peace* . . .



# generative story in LDA

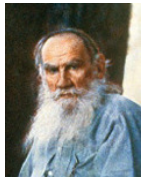
- . . . Tolstoy first selected a topic composition randomly

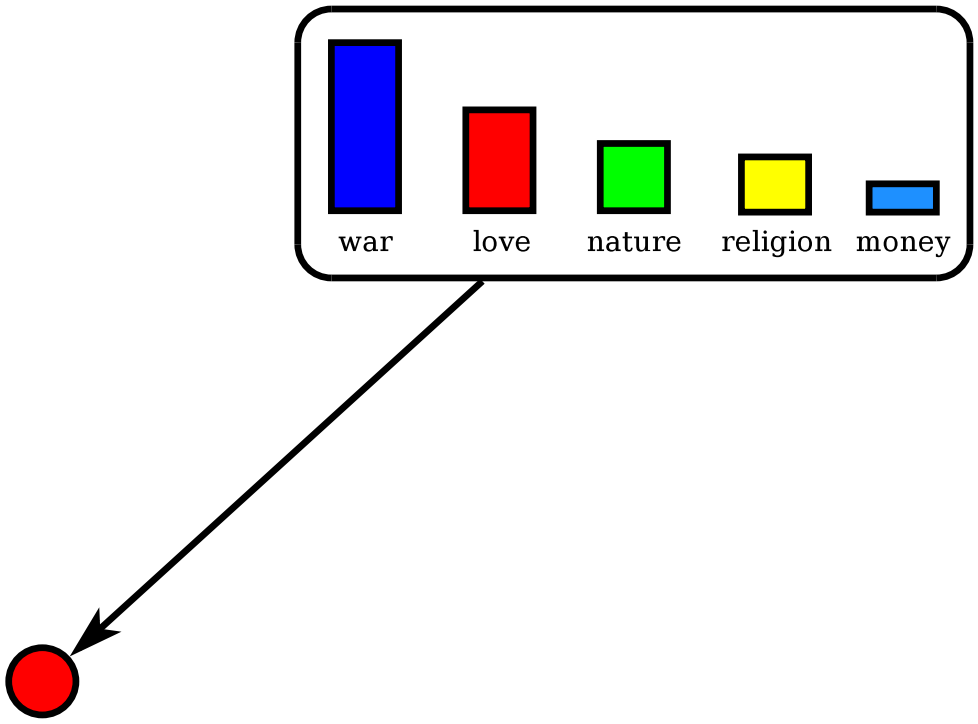




# generative story in LDA

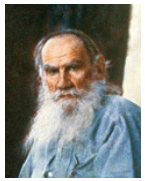
- he then selected a topic for the first word

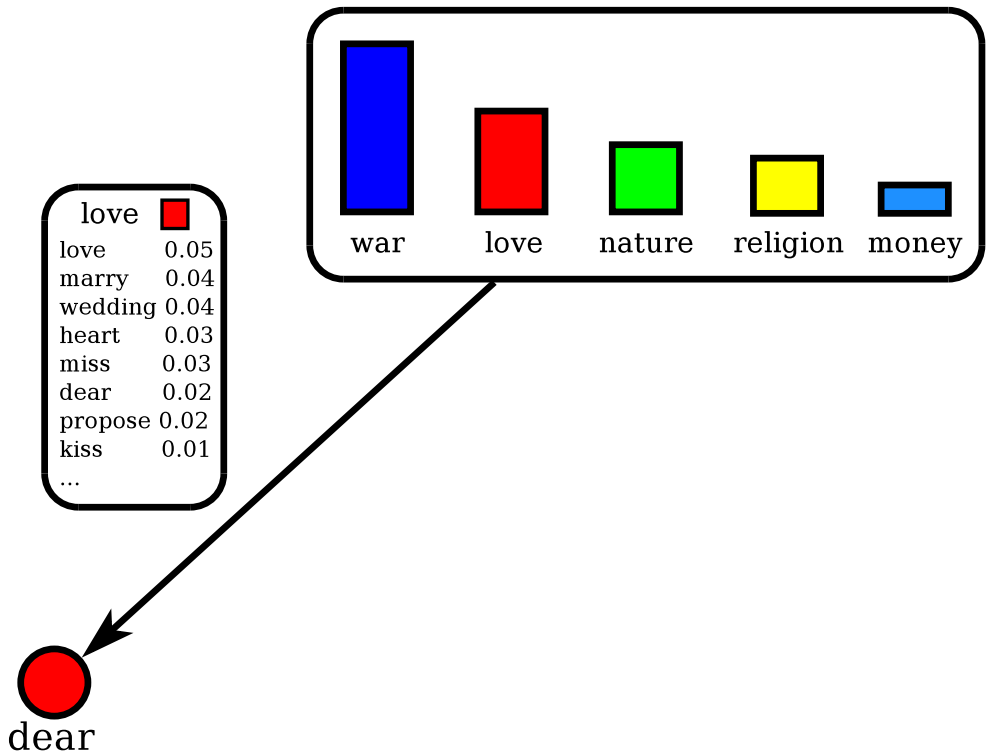




# generative story in LDA

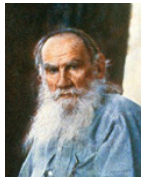
- . . . and then decided which word to write

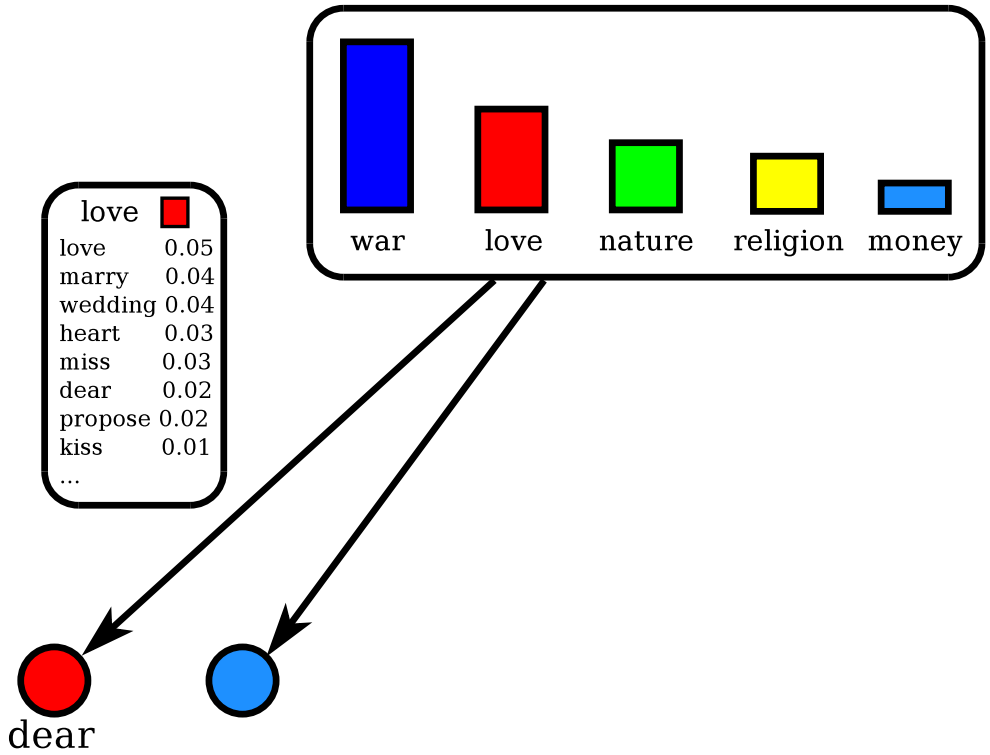




# generative story in LDA

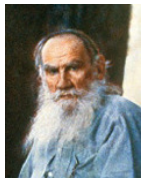
- he then selected the topic for the second word

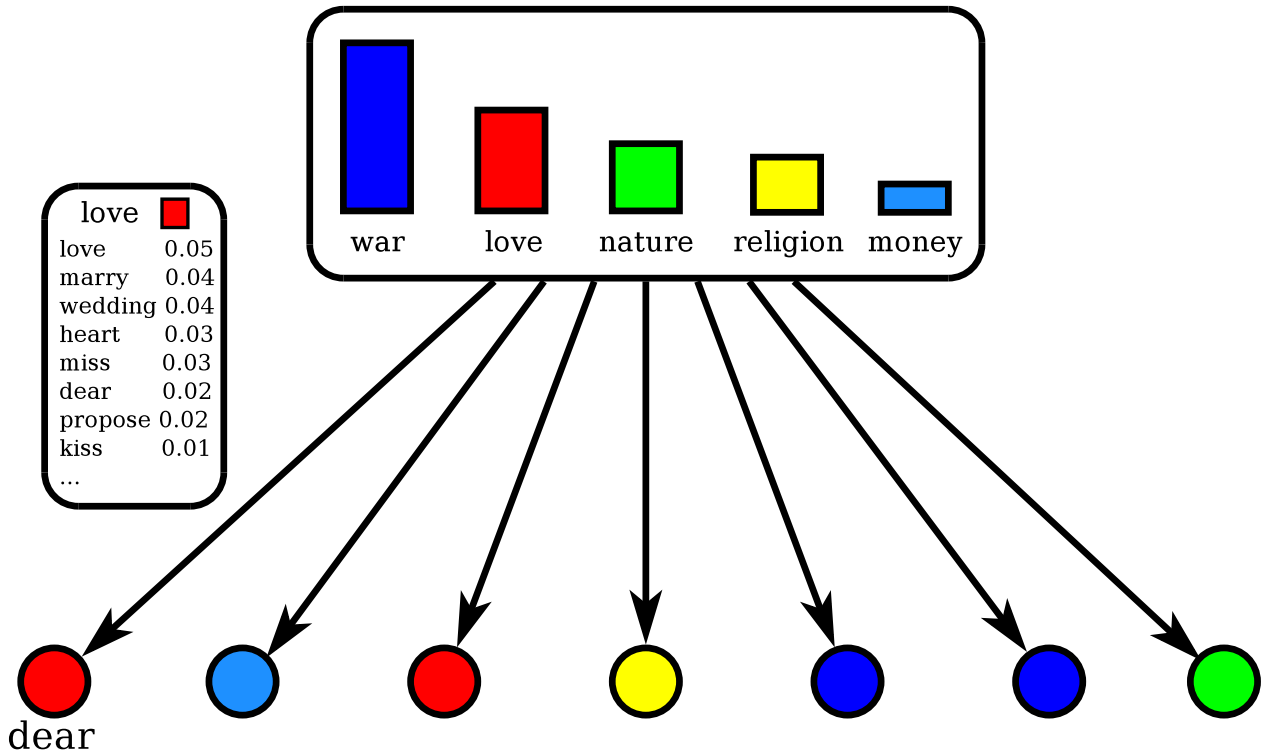




# generative story in LDA

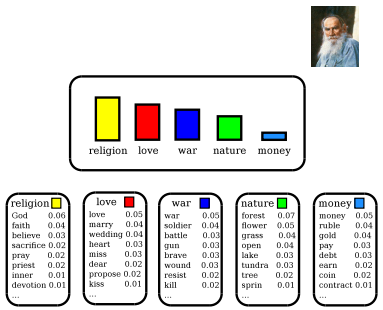
- . . . and so on



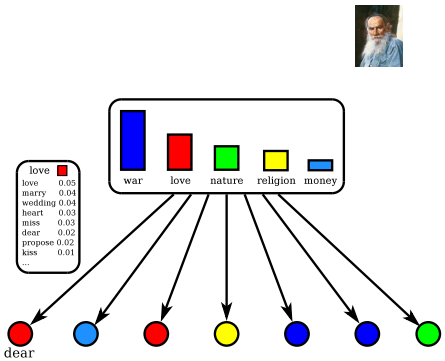


# estimation in topic models

- given a corpus of documents, the LDA estimation process tries to "reverse-engineer" the generative model
- reconstruct the topics and their overall distributions




- reconstruct the composition of topics in each document



# LDA implementation

- implementing LDA is a bit complex; there are two typical approaches
  - an EM-like procedure called variational inference
  - a randomized algorithm called Gibbs sampling
- there are a number of software libraries containing LDA:
  - gensim (Python): `http://radimrehurek.com/gensim`
  - Mallet (Java): `http://mallet.cs.umass.edu`
  - Blei's homepage:

    `http://www.cs.princeton.edu/~blei/topicmodeling.html`
  - Mahout (built on Hadoop, for large-scale processing):

    `https://mahout.apache.org`
  - **NB**: LDA in scikit-learn is something else

# example: extracted topics in the Swedish Wikipedia

- "**plant species**" (0.044): *ingå familj släkte art beskriva lista underart Inga svamp först namn rike gälla division sporsäck...*
- "**famous Swedes**" (0.031): *svensk född Stockholm år död Sverige Göteborg ledamot samt ordförande universitet ...*
- "**Anglo-Saxons**" (0.027): *amerikansk född of the New USA år död York John brittisk University London England William...*
- "**film and TV**" (0.018): *film serie spela the skådespelare amerikansk avsnitt roll tv-serie år program TV-serie...*
- "**sport**" (0.014): *spela lag match klubb säsong vinna spelare år fotboll mål division serie liga final turnering landslag...*

# example: topics for some Swedish Wikipedia articles

- **Gothenburg**: “city buildings” 0.70, “Swedish bureaucracy” 0.15, “culture” 0.08, “geography” 0.05
- **jazz**: “music” 0.50, “language and theory” 0.24, “records” 0.18, “Anglo-Saxons” 0.05
- **natural language processing**: “language and theory” 0.60, “computers” 0.24
- **Python**: “programming” 0.55, “computers” 0.42
- **Silvio Berlusconi**: “politics” 0.46, “Catholicism and southern Europe” 0.10, “film and TV” 0.08, “commerce” 0.07
- **Zlatan Ibrahimovic**: “sport” 0.88, “family” 0.05, “commerce” 0.02

# overview

# the remainder

- **March 12**: machine translation (with Prasanth)
- March 17 and 19: VG assignment lab sessions (and catchup)