# Topic Modeling of Document collections

Jerónimo Arenas-García, Vanessa Gómez-Verdejo, Jesús Cid-Sueiro

Universidad Carlos III de Madrid

*jeronimo.arenas@uc3m.es*

March 24, 2021

# Contents

# Topic models

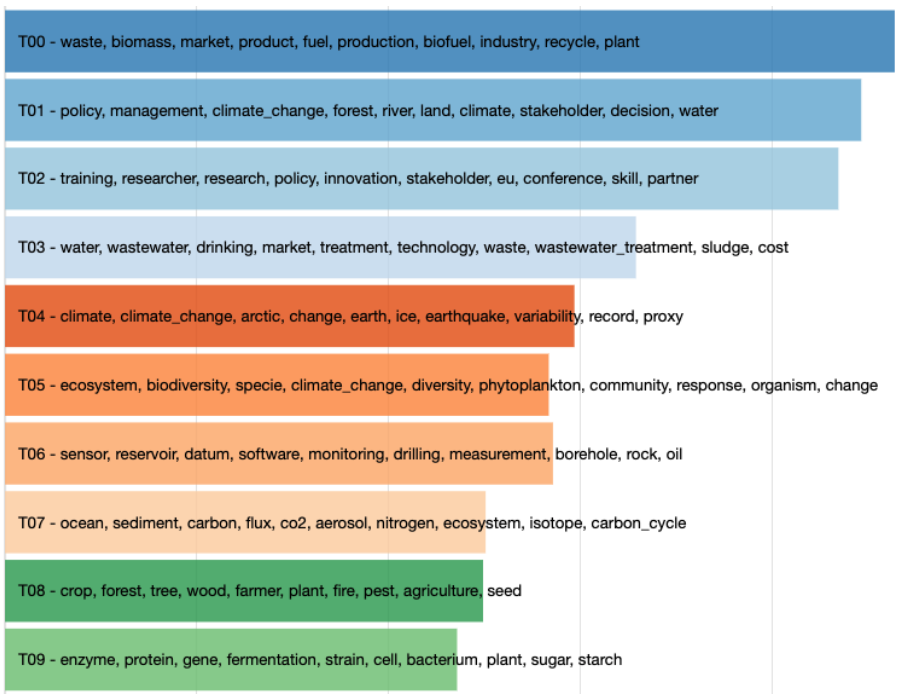
- Topic Models attempt to uncover the underlying semantic structure of a document corpus by identifying recurring patterns of terms (topics).

- Topic models are models for bags-of-words:
  - do not parse sentences
  - do not care about word order, and
  - do not "understand" grammar or syntax

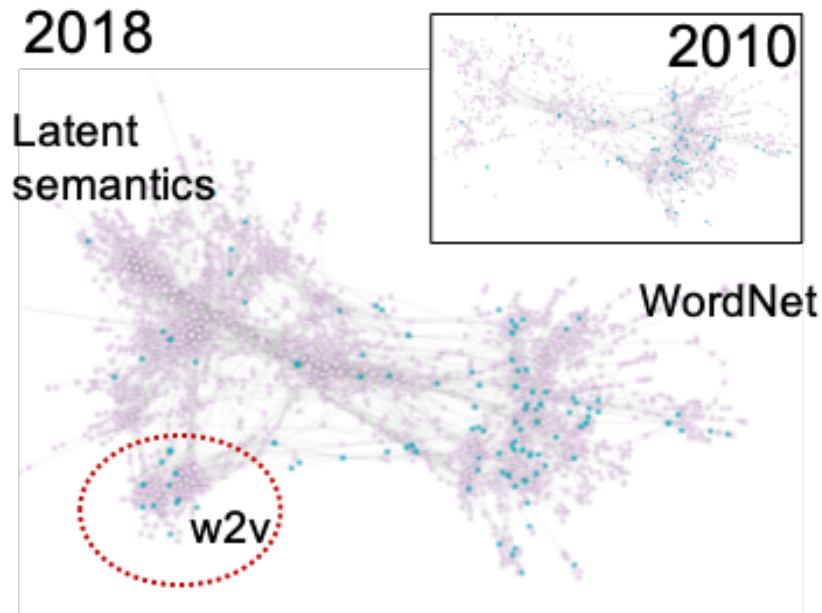
# Topic models

- Topic models are useful on their own to build visualizations and explore data. They are also very useful as an intermediate step in many other tasks.

## 1. Identification and characterization of the main topics of a Document Collection




## 2. Obtaining a (semantic) vector representation of each document in the collection(s)



# Topic modeling tools

- **Gensim** is an NLP toolbox Developed by Radim Rehurek, and specialized in topic modeling and semantic analysis of texts. It includes:
  - Tools for BoW codification of documents
  - Tools for TF-IDF representation of documents
  - **Latent Semantic Indexing (LSA/LSI)**
  - **Latent Dirichlet Allocation (LDA)**
  - Dynamic topic modeling (incomplete)
  - Word2Vec tools for mapping words in a vector semantic space
- In the block we will rely on David Blei's LDA algorithm that rely on the BoW representations. Other available Python implementations are:
  - LDA toolbox
  - Scikit learn implementation
  - GraphLab (based on collapsed Gibbs sampling)
- **MALLET** contains a well-known Java implementation of LDA

# Topic model visualizers

Given the extended use of topic models, some researchers have also publish tools for visualizing the results of topic models.

- dfr-browser: `https://github.com/agoldst/dfr-browser`
  Demo: `https://agoldst.github.io/dfr-browser/demo/`
- **pyLDAvis:** `https://github.com/bmabey/pyLDAvis`
- Topic model visualization Engine (TMVE):
  `https://github.com/ajbc/tmve-original`
  Demo: `http://www.princeton.edu/~achaney/tmve/wiki100k/browse/topic-presence.html`

# Contents

# Latent Semantic Indexing

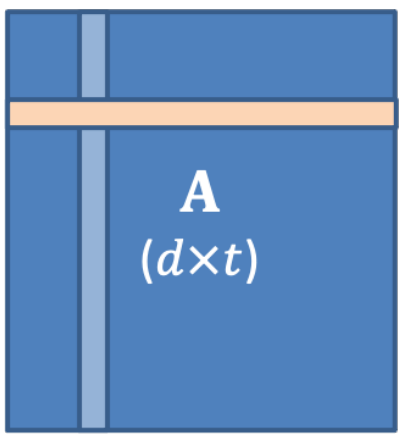
- It transforms documents from either
  - bag-of-words, or
  - (preferably) TFIDF document representation

  into a latent space of a lower dimensionality (number of topics)
- LSI is able to identify correlations among semantically related terms that are latent in a collection of text documents
- When used to measure similarity among documents, LSI overcomes two of the most problematic constraints of Boolean keyword queries: synonyms and polysemy.

# Latent Semantic Indexing

The starting point of LSI is the TF-IDF matrix, $\mathbf{A}$, with size $(d \times t)$,

- $d$ is the number of documents
- $t$ is the number of terms in the vocabulary




- $i$th row: TFIDF representation of document $i$
- $i$th column: associated to term $i$ in the vocabulary

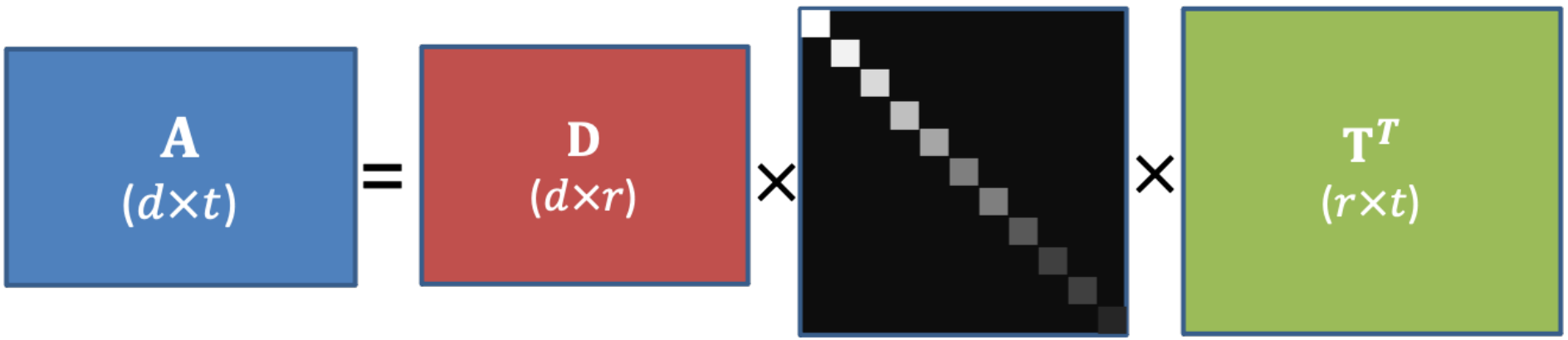
# Latent Semantic Indexing

LSI computes the term and document vector spaces by means of the singular value decomposition (SVD) of matrix **A**, with rank $r$, into the product of 3 matrices:

$$\mathbf{A} = \mathbf{DST}^\top$$

- $\mathbf{D}(d \times r)$: Unitary document-concept matrix ($\mathbf{D}^\top\mathbf{D} = \mathbf{I}_r$)
- $\mathbf{T}(t \times r)$: Unitary term-concept matrix ($\mathbf{T}^\top\mathbf{T} = \mathbf{I}_r$)
- $\mathbf{S}(r \times r)$: Diagonal matrix of singular values

$$(s_0 > s_1 > \cdots > s_{r-1} > 0)$$

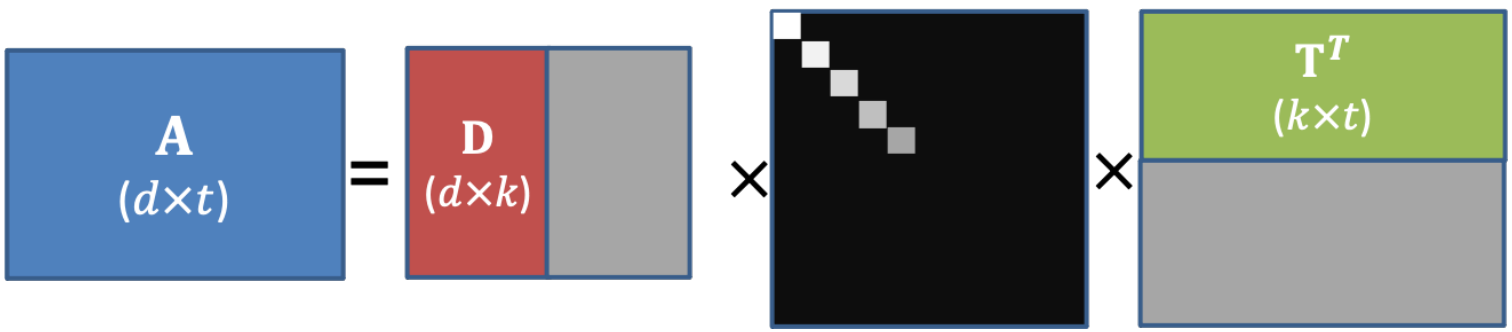


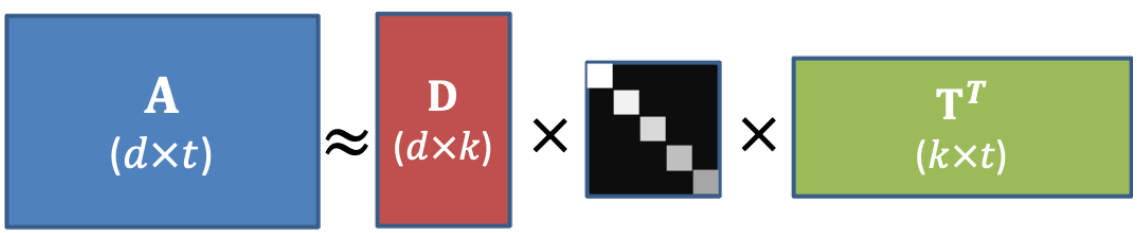

(Note that elements in **D** and **T** can have any sign.)

# Latent Semantic Indexing

LSI approximates **A** using a reduced number $k$ of concepts (the topics in LSI), by ignoring the smallest values.

$$\mathbf{A} \approx \mathbf{D}_k \mathbf{S}_k \mathbf{T}_k^\top$$

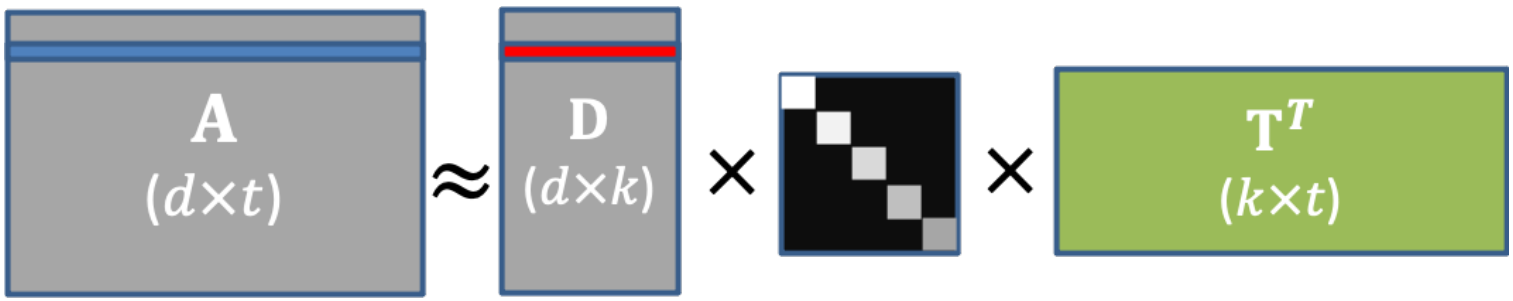


# Latent Semantic Indexing

Note that the $n$th row of $\mathbf{A}$ depends only on the $n$th row of $\mathbf{D}$

- The $n$th row of $\mathbf{D}$ is the "latent" representation of the $n$th document
- The TFIDF representation of the $n$th document is approximated a linear combination of columns in $\mathbf{T}$ (rows of $\mathbf{T}^\top$)
- Each column of $\mathbf{T}$ (of size $t \times 1$) is the characterization of a different LSI "topics"




## Intuitive explanation

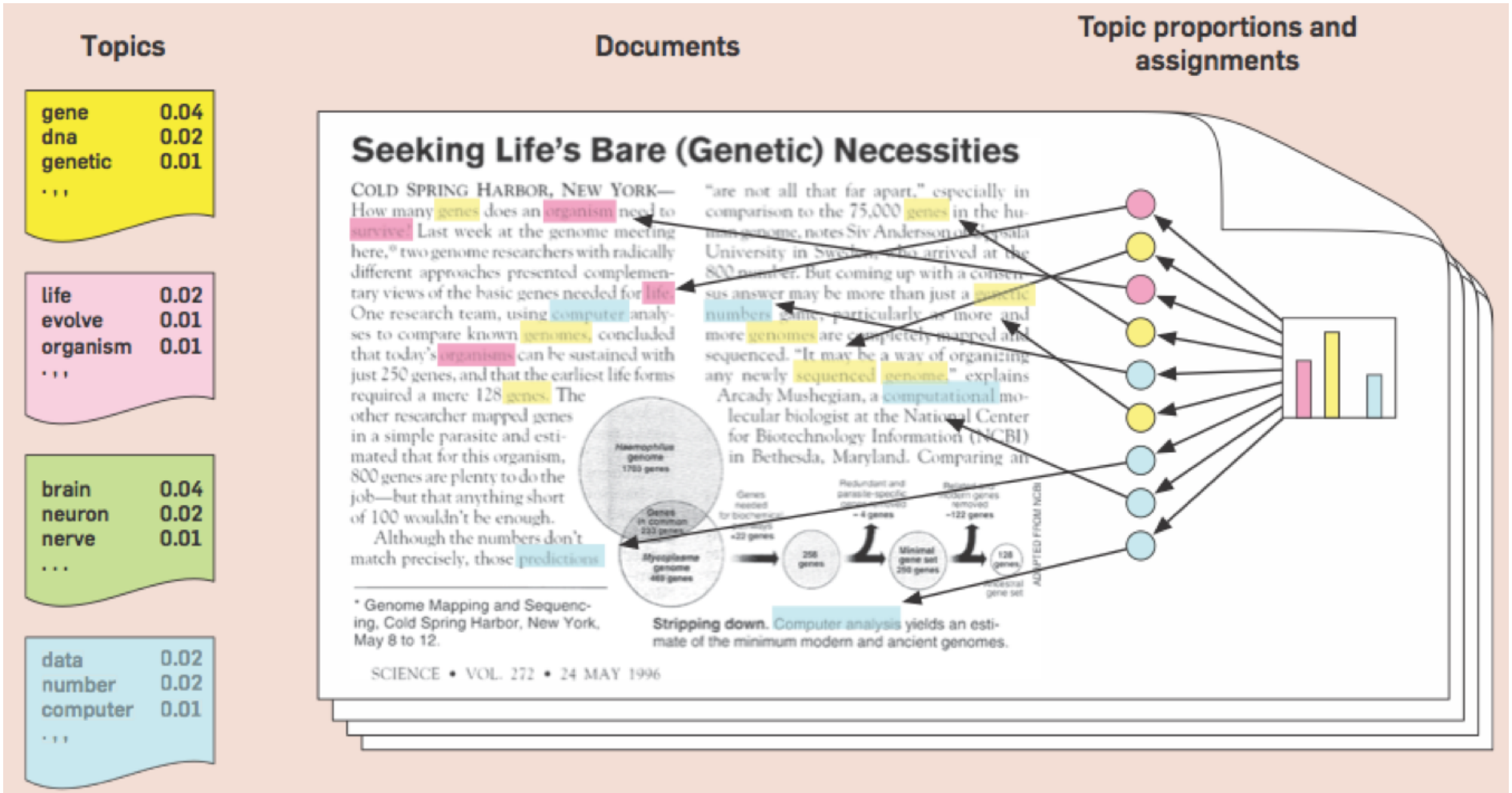
Since the SVD provides the best reduced-rank approximation of $\mathbf{A}$, this implies that the topic matrix will be adjusted accordingly, aligning the topic vectors with the dominant themes in the corpus.

# Contents

# Latent Dirichlet Allocation

- LDA is another transformation from bag-of-words into a latent topic space of lower dimensionality
- LDA is a probabilistic extension of LSI. It assumes a generative model where
  - topics are characterized by probability distributions over words
  - documents are characterized by probability distributions over topics
- These distributions are inferred automatically from a training corpus

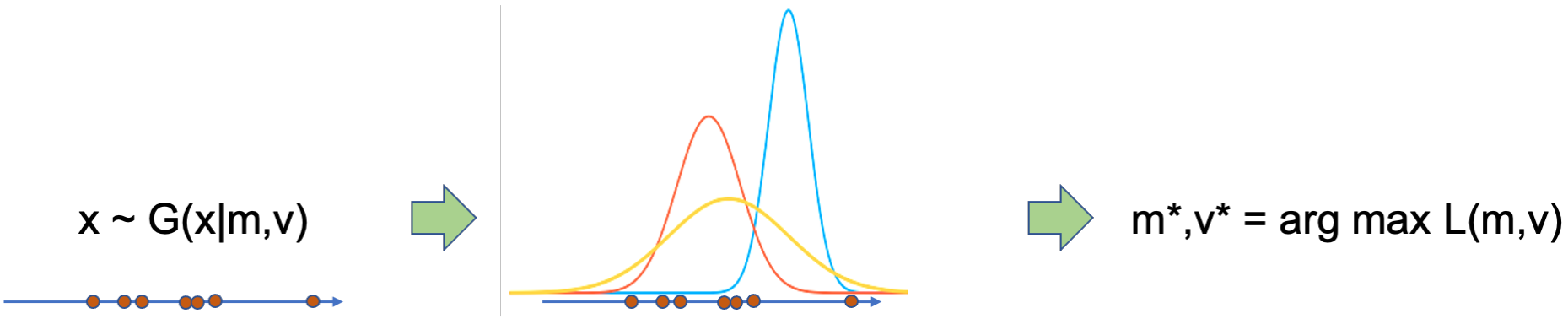
# Latent Dirichlet Allocation




(Image taken from `http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf`)

# Maximum Likelihood and Maximum a Posteriori Estimation

1. Propose a parameterized generative model
2. For the available observations, write down the likelihood function, i.e., the joint probability of observations for a given set of parameter values
3. Find the parameter values that maximize the likelihood function




$$L(m,v) = p(x_0|m,v)\, p(x_0|m,v) \ldots p(x_{K-1}|m,v)$$

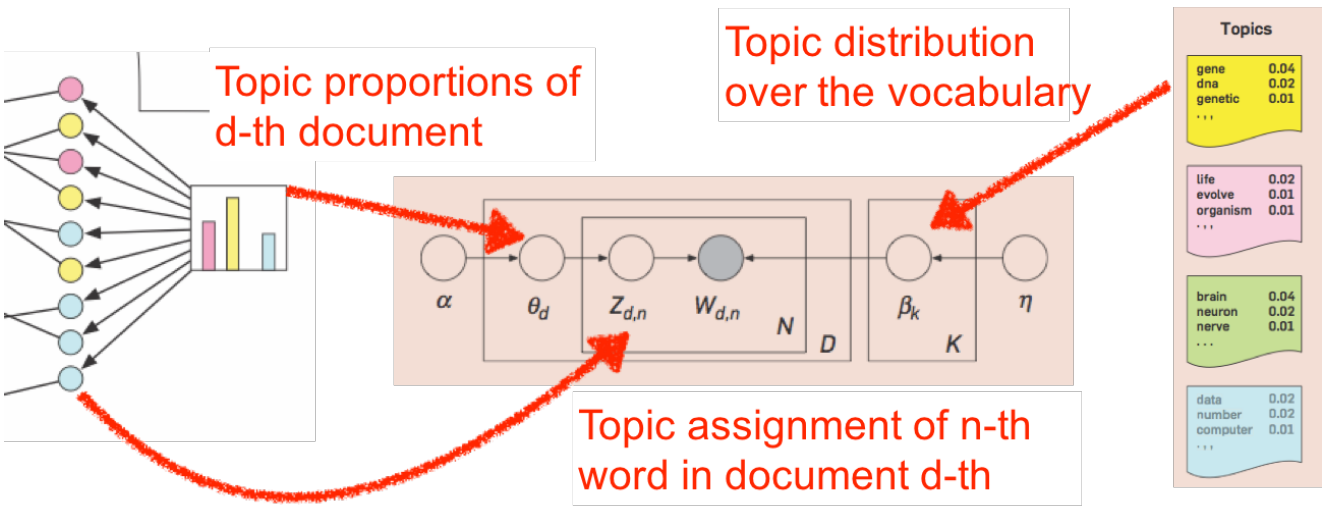
In maximum *a posteriori*, we assume some *prior* distribution for the unknown parameters, and maximize their *a posteriori* distribution, taking into account that:

$$posterior \propto (likelihood \times prior)$$

# Understanding LDA

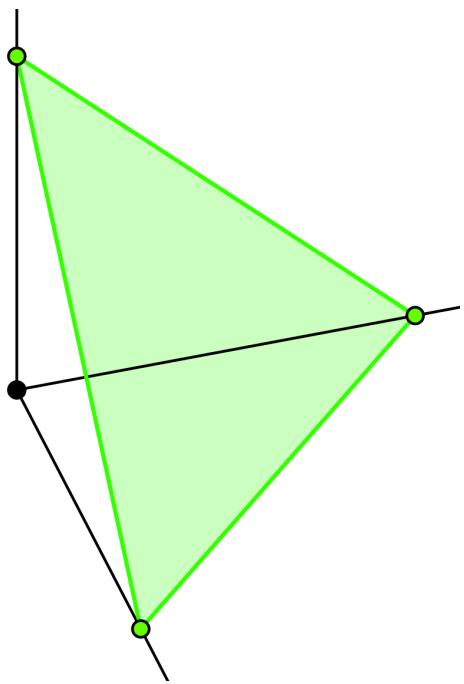
LDA is also based on MAP estimation of the parameters of a generative probabilistic model for document production:

- Documents have been generated according to some probability model with some unknown parameters and certain hidden variables
- Our observations are the documents
- The corpus data is used to infer the topic structure (hidden variables) from the words of documents (observed variables)



# The Dirichlet Distribution

The Dirichlet distribution is frequently used to generate probability vectors. For instance, when we sample a 3D Dirichlet distribution, we will obtain triplets of positive numbers that sum up to 1.



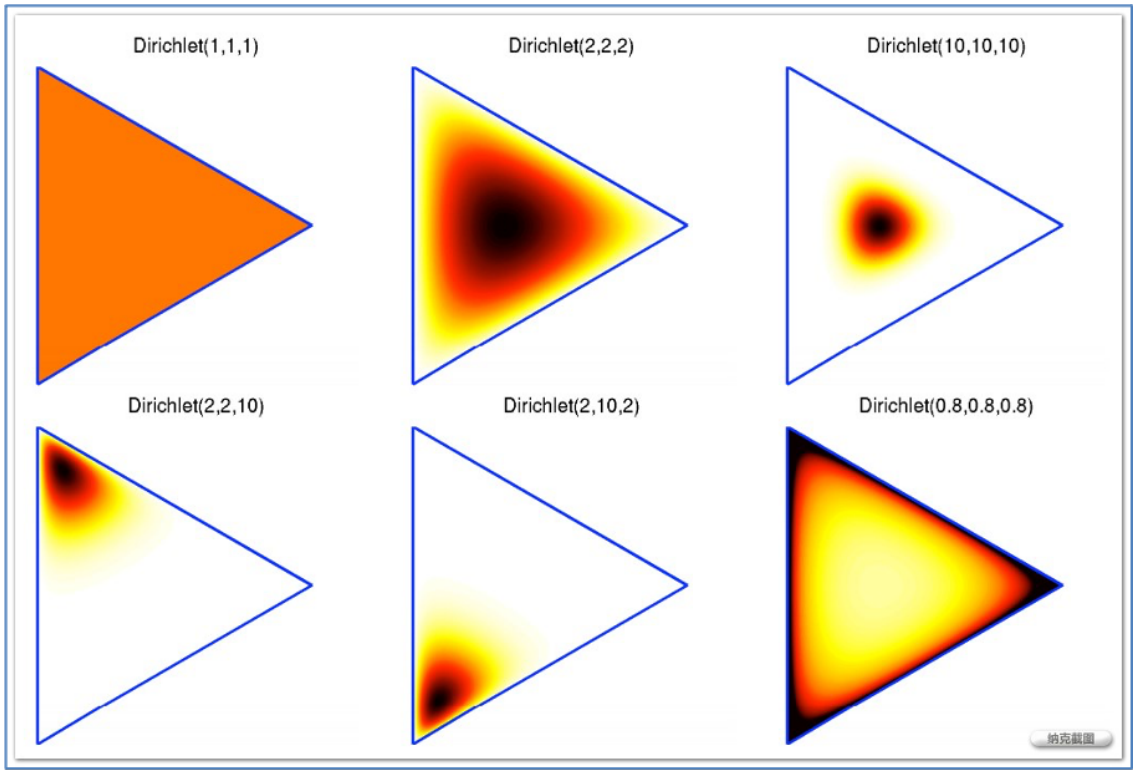
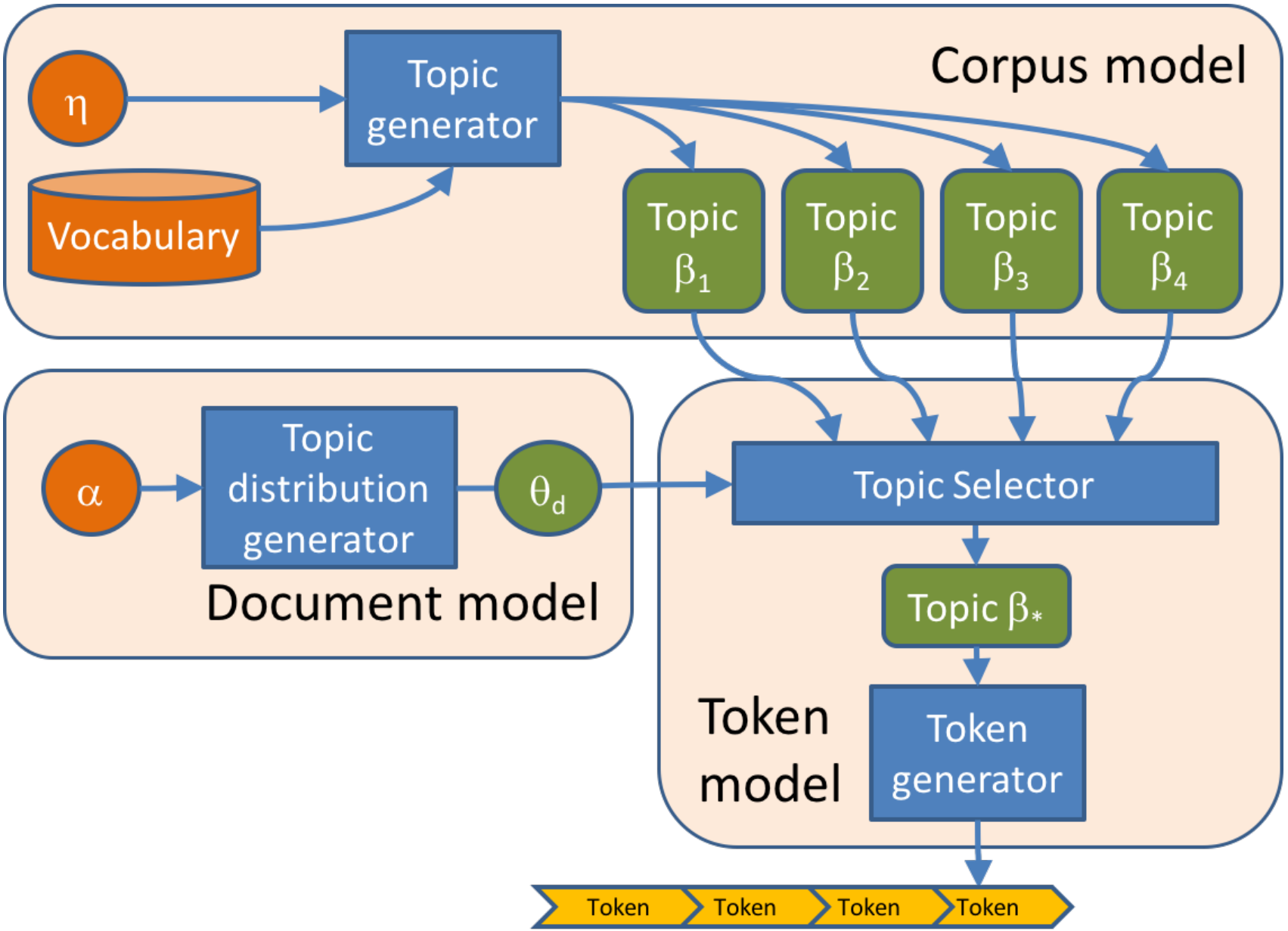



Image taken from http://www.cnblogs.com/huashiyiqike/articles/3232082.html

# LDA generative model



# LDA generative model (II)

## Topic generator

- Generates tokens distributions by means of a Dirichlet distribution with concentration parameter $\eta$.
- Each topic is therefore characterized by a vector $\boldsymbol{\beta}_t$ that indicates the term distribution for the given topic
- Small $\eta$ implies that topic distributions will be concentrated in a few distinct tokens (only a few non-zero components in each $\boldsymbol{\beta}_t$).

## Document generator

- For each document, generate its topic distribution by sampling a Dirichlet distribution with concentration parameter $\alpha$.
- Small $\alpha$ implies that for most documents only a few topics will be relevant.
- For each word in the document:
  1. Select a topic according to a multinomial distribution with parameters given by the topic distribution for the document
  2. Select a word according to a multinomial distribution with parameters given by the word distribution for the selected topic

# LDA optimization

- The generative process for LDA corresponds to the following joint distribution of the hidden and observed variables

$$p(\boldsymbol{\beta}_{1:K}, \boldsymbol{\theta}_{1:D}, \mathbf{z}_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\boldsymbol{\beta}_i) \prod_{d=1}^{D} p(\boldsymbol{\theta}_d) \left( \prod_{n=1}^{N_d} p(z_{d,n}|\boldsymbol{\theta}_d) p(w_{d,n}|z_{d,n}, \boldsymbol{\beta}_{1:K}) \right)$$

- Goal: computing the conditional distribution of the topic structure given the observed documents

$$p(\boldsymbol{\beta}_{1:K}, \boldsymbol{\theta}_{1:D}, \mathbf{z}_{1:D}|w_{1:D}) = \frac{p(\boldsymbol{\beta}_{1:K}, \boldsymbol{\theta}_{1:D}, \mathbf{z}_{1:D}, w_{1:D})}{p(w_{1:D})}$$

- The denominator is the probability of seeing the observed corpus under any topic model. It can be computed by summing the joint distribution over every possible hidden topic structure, so its computation is not feasible.
- LDA implementations are based on:
  - Variational Bayes implementations
  - Sampling-based algorithms (typically, collapsed Gibbs-sampling with the goal of estimating word-assignments, $\mathbf{z}_{1:D}$)

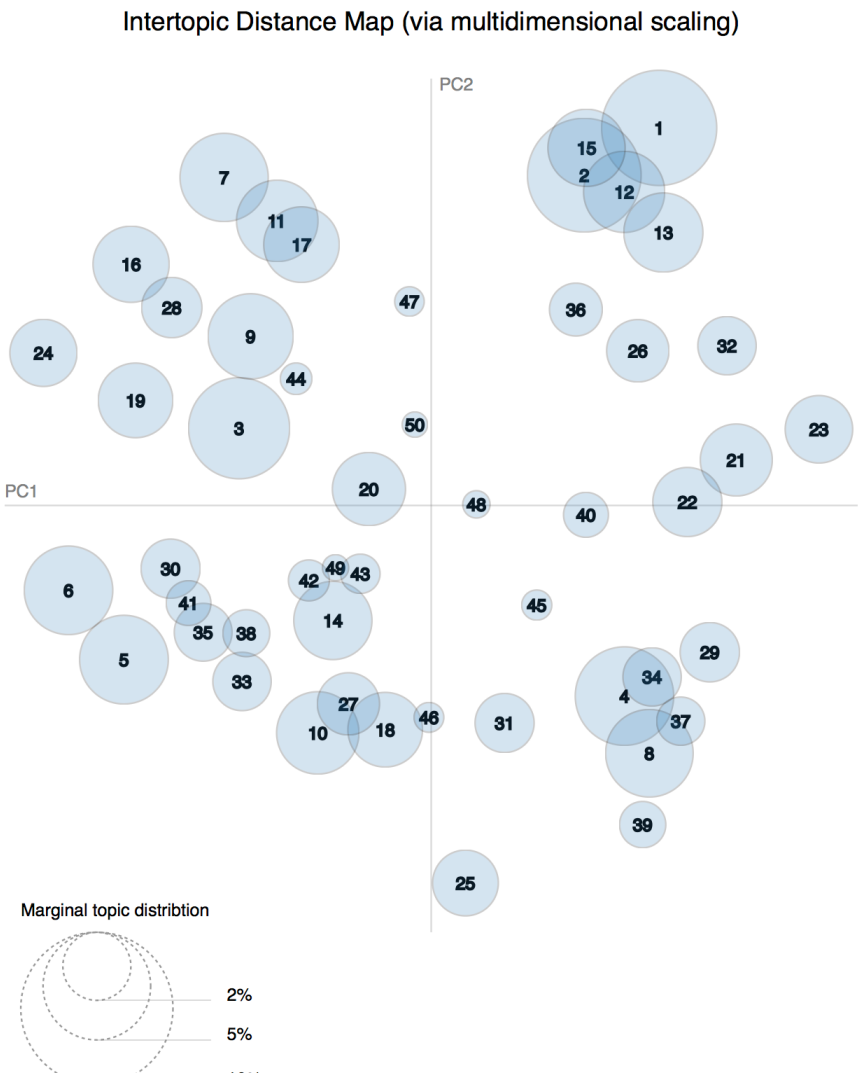
# LDA optimization (II)

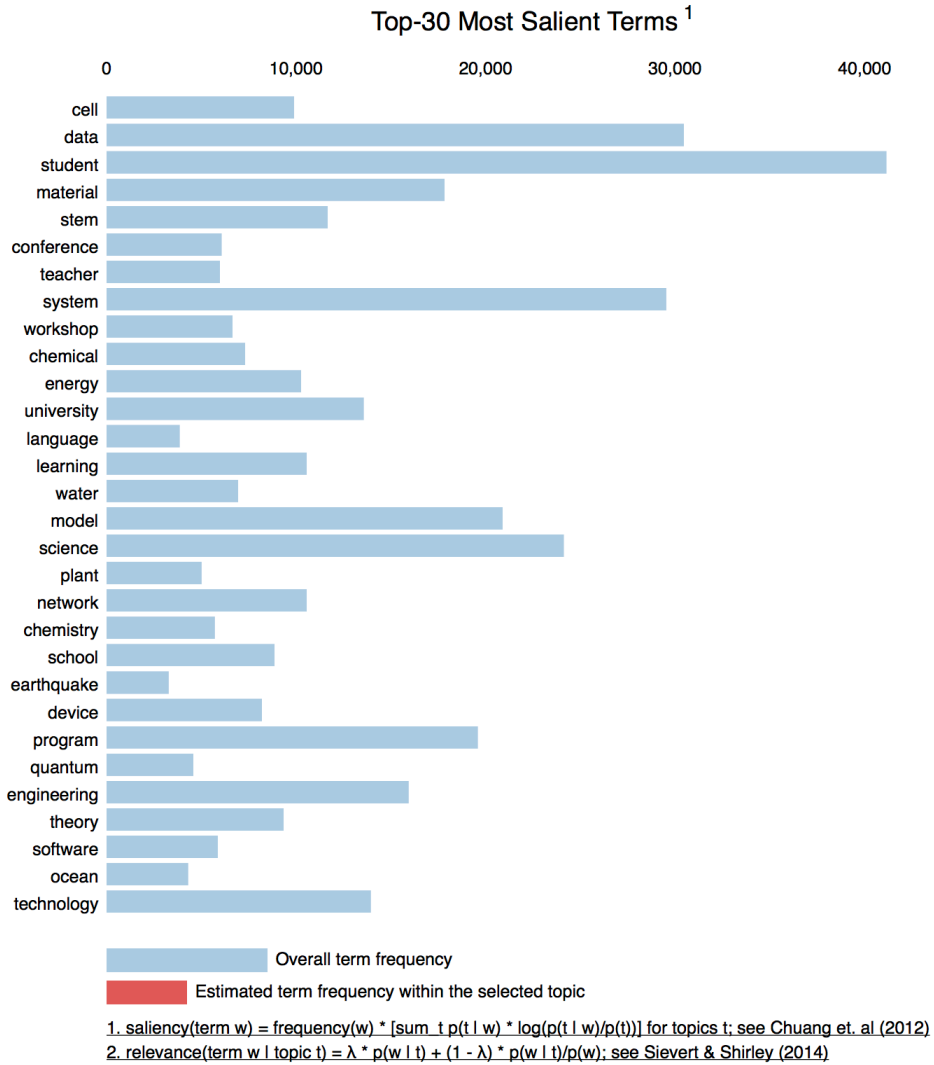
After optimization, we will be mostly interested on the outcome of the Dirichlet distributions:

- $\boldsymbol{\theta}_{1:D}$ provide the topic representation for each document in the collection.
- $\boldsymbol{\beta}_{1:K}$ provide the probability distribution over the vocabulary of each identified topic.
- We will ignore the estimated assignments of each word $(z_{d,n})$

The average of the topic representation of all documents $(\boldsymbol{\theta}_{1:D})$ is an estimation of the relative size of topics in the corpus.

# Topic visualization using pyLDAvis



# Contents

# Document BoW and TFIDF representation

```
from gensim import corpora
from gensim import models

D = corpora.Dictionary(docs)
corpus_bow = [D.doc2bow(doc) for doc in docs]

tfidf = models.TfidfModel(corpus_bow)
corpus_tfidf = tfidf[corpus_bow]
```

# Latent Semantic Indexing

```
# Initialize an LSI transformation.
# On real corpora, target dimensionality of
# 200-500 is a resonable first guess

lsi = models.LsiModel(corpus_tfidf,
    id2word=D, num_topics=200)

corpus_lsi = lsi[corpus_tfidf]

#It allows incremental updates
lsi.add_documents(another_tfidf_corpus)
corpus_lsi = lsi[corpus_tfidf]
```

# Latent Semantic Indexing (II)

```
# Analyzing the topics

lsi.print_topics(2)

topic #0(1.594): -0.703*"trees" + -0.538*"graph" +
                 -0.402*"minors" +...
topic #1(1.476): -0.460*"system" + -0.373*"user" +
                 -0.332*"eps" +....

# Analyzing document representation

print(corpus_lsi[0])

# "The intersection graph of paths in trees"
[(0, -0.877), (1, -0.168)]
```

# Latent Dirichlet Allocation

```
# Create an LDA transformation
lda = models.LdaModel(corpus_bow,
    id2word=D, alpha='auto', num_topics=20)

# Analize topics
lda.print_topics(topics=2, topn=5)

# get topic probability distribution for a document
print(lda[doc_bow])
```

# Easter Assignments

1. Select and acquire a corpus of your own (default selection: ACL)
2. Homework to be published: Notebook on Sentiment Analysis using SpaCy
3. Review this presentation before next session