Exploring and understanding documental databases with topic models and graph analysis

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

Universidad Carlos III de Madrid

jeronimo.arenas@uc3m.es

August 31, 2017

Contents

Natural Language Processing (NLP)

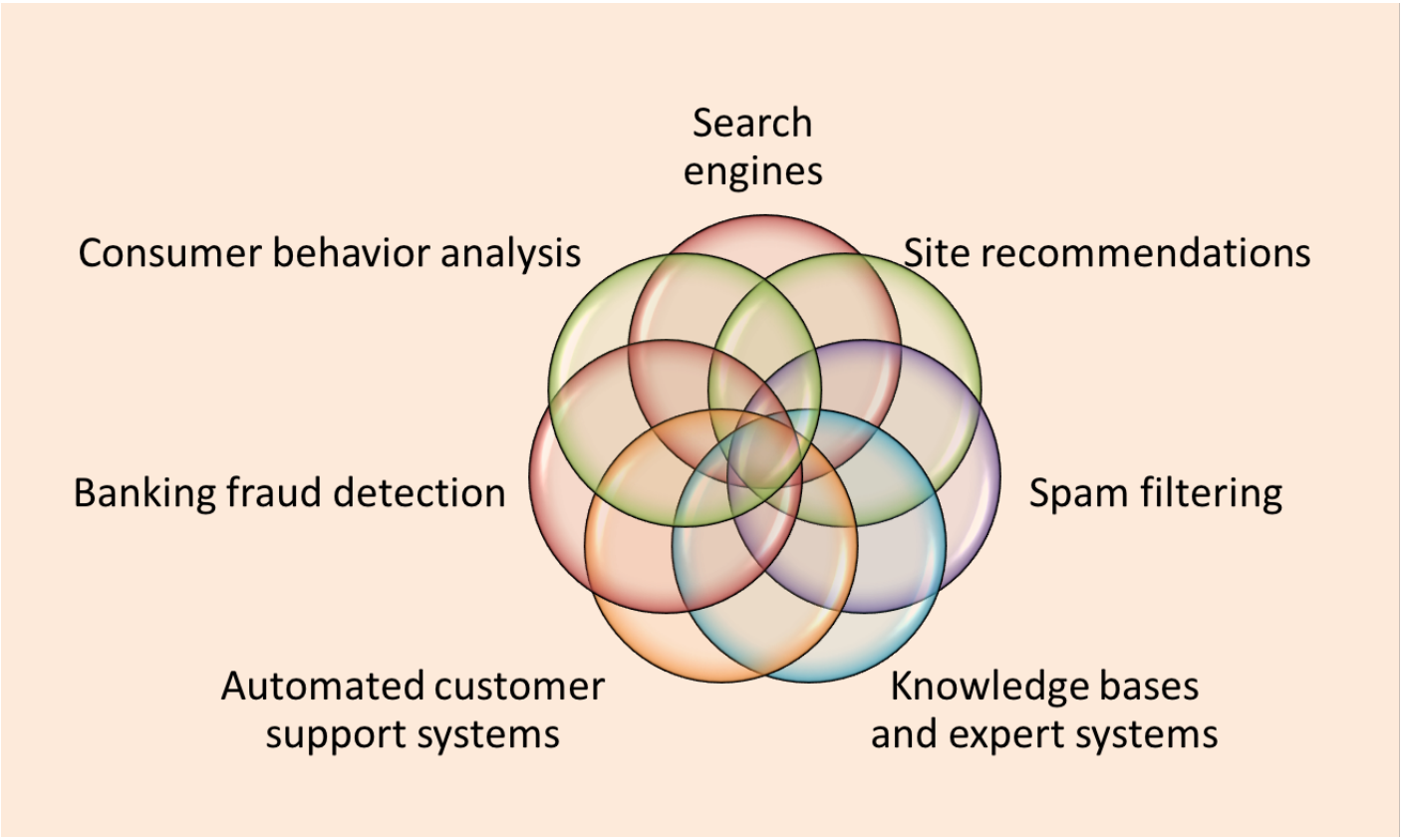
Definition

- Computer aided text analysis of human language
- The goal is to enable machines to understand human language and extract meaning from text

This is a field of study which falls under the intersecting interests of (at least):

- machine learning
- computational linguistics

Natural Language Processing Applications



Here, we will pay attention to tools for the unsupervised **semantic** organization of documental databases.

NLP challenges to machines

Generic difficulties

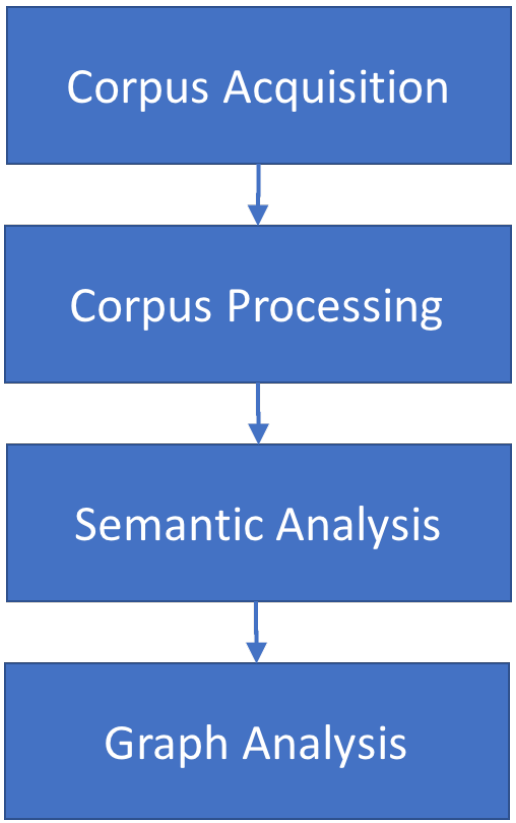
- Natural Language can be highly **ambiguous**
- **Disambiguation** of polysemic words
- **Sentiment Analysis**
- **Intention inference**: sarcasm, humor, etc
- Differentiate main from secondary topics
- Incorporate domain-expert knowledge into automatically inferred statistical models

Specific applications

- Text summarization
- Machine dialog systems

Module overview

We will study the topics for research funded by NSF during recent years, and infer thematic relations among institutions. We will



- Discuss possibilities for corpus acquisition
- Present some of the most relevant text preprocessing techniques in the NLTK
- Present theoretical foundations and implementations of topic modeling algorithms
- Use the semantic representations to infer graphs among entities, and visualize them using Gephi

Python libraries we will use

- The Natural Language Processing Toolkit (NLTK)
  `http://www.nltk.org`
  Best for education and research, but not optimized for SW production
  - Corpora
  - Entity recognition
  - POS tagging
  - Stemming and lemmatization
  - Stopword lists
  - + other text preprocessing and homogeneization tools
- Gensim
  `https://radimrehurek.com/gensim/`
  Library specialized on topic modeling and document similarity analysis
  - Topic models: LDA, LSI
  - Word2vec and Doc2vec
  - Pagerank

Contents

Text sources

There are many possibilities for acquiring interesting textual databases

- Web content (web pages, twitter, blogs, etc)
  - Crawlers
  - Available APIs (e.g., wikipedia, arXiv)
- Local document collections
- Available corpus

Some applications examples

- Use job portals for modeling the job market in a country
- Analyze innovation from patent descriptions
- Categorize apps from their textual description in App stores
- **Analyze public funding topics using projects abstracts**

National Science Foundation data

We will work with project description and metadata taken from the National Science Foundation website (`http://www.nsf.org`). The information is encoded in XML format, as shown in the example below:

```
<?xml version="1.0" encoding="UTF-8"?>
<rootTag>
<Award>
<AwardTitle>Novel States in Spin-Orbit-Coupled and Correlated Materials</AwardTitle>
<AwardEffectiveDate>08/01/2016</AwardEffectiveDate>
<AwardExpirationDate>12/31/2016</AwardExpirationDate>
<AwardAmount>306810</AwardAmount>
<AwardInstrument>
<Value>Continuing grant</Value>
</AwardInstrument>
<Organization>
<Code>03070000</Code>
<Directorate>
<LongName>Direct For Mathematical &amp;amp; Physical Scien</LongName>
</Directorate>
<Division>
<LongName>Division Of Materials Research</LongName>
</Division>
</Organization>
<ProgramOfficer>
<SignBlockName>Tomasz Durakiewicz</SignBlockName>
</ProgramOfficer>
<AbstractNarration>Non-technical Abstract:&lt;br/&gt;Modern condensed matter physics research has produced novel mater
ials with fundamental properties that underpin a remarkable number of cutting-edge technologies. It is now generally a
ccepted that novel materials are necessary for critical advances in technologies and whoever discovers novel materials
generally controls the science and technology of the future. Transition metal oxides have attracted enormous interest
within both the basic and applied science communities.  However, for many decades, the overwhelming balance of effort
was focused on the 3d-elements (such as iron, copper, etc.) and their compounds; the heavier 4d- and 5d-elements (such
as ruthenium, iridium, etc., which constitute two thirds of the d-elements listed in the Periodic Table) and their com
```

NSF dataset construction

Relevant files

- Zip files under the 'data' directory (contain compressed projects information and metadata)
- Python notebook '02901.ipynb'

Exercise

- Go through subsections 1.1.1. and 1.1.2. that explain the basics of working with XML files in python
- Complete the exercise described in subsection 1.1.3. You have to implement a function that extracts the following information for each project:
  - Text data (for semantic analysis): Title, Abstract
  - Metadata: Budget, Starting Year, Institution
- Go through the rest of Chapter 1 to see how the whole dataset is constructed and to understand its structure

Contents

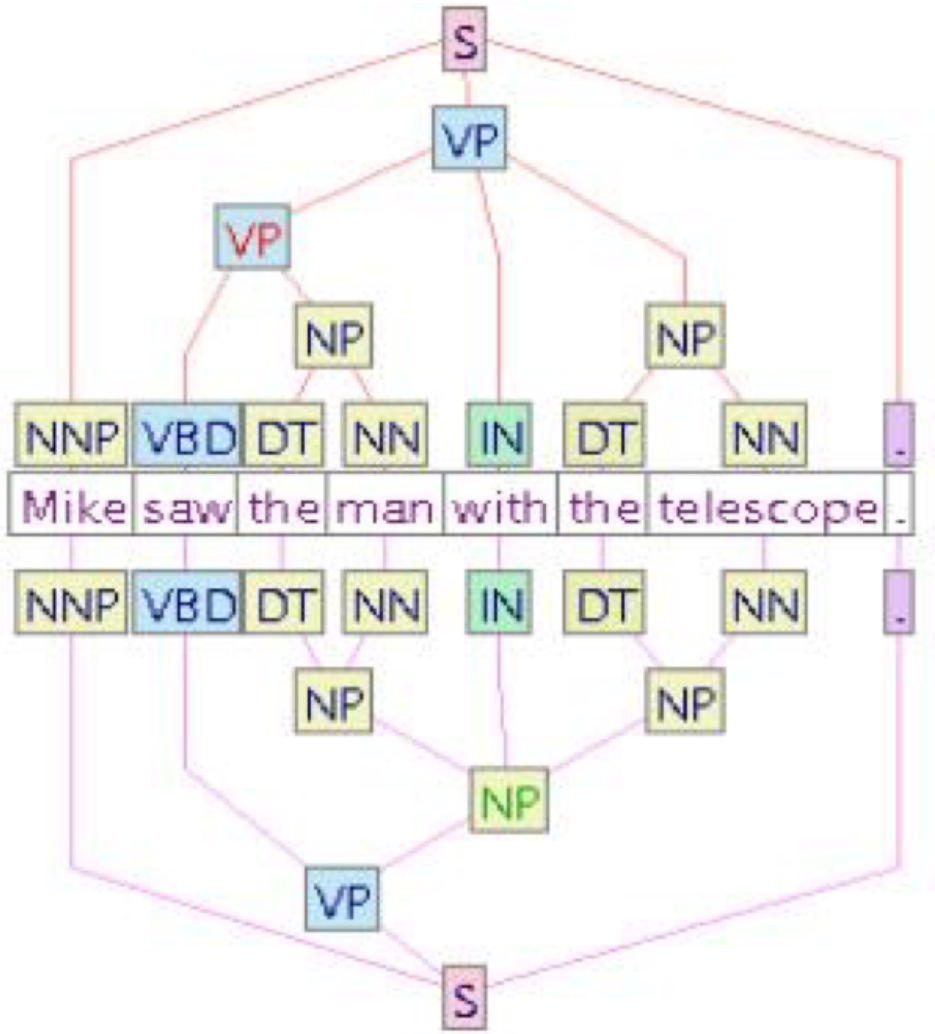
Document representation for Machine Learning

- ML algorithms process numbers, not words
- We need to transform text into meaningful numbers
- Bag-of-Words (BoW) representation: Count number of appearances of each word (in the vocabulary of all documents) in a given document
- In order to have a useful representation, some preprocessing steps are normally required

NLP with Python

NLTK: A package that provides:

- Basic classes for representing data relevant to natural language processing
- Functions for performing common NLP tasks (parsing, tokenization, etc)
- Corpuses (text collections, stopword lists, etc)



NLTK modules

- corpora: a package containing collections of example text
- **tokenize**: functions to split text strings into basic elements
- probability: for modeling frequency distributions and probabilistic systems
- **stem**: package of functions to stem words of text
- **wordnet**: interface to the WordNet lexical resource
- chunk: identify short non-nested phrases in text
- etree: for hierarchical structure over text
- **tag**: tagging each word with part-of-speech, sense, etc.
- parse: building trees over text - recursive descent, shift-reduce, probabilistic, etc.
- cluster: clustering algorithms
- draw: visualize NLP structures and processes
- contrib: various pieces of software from outside contributors

Tokenization

- The goal is to find the basic elements (tokens) of a given string.
- Python `split()` function makes a list from a string using a given substring as a separator
- NLTK tokenizers are better suited to this task, and more efficient since they use regular expressions.

Example

```
≫ sentence = 'Hello, world.'
≫ sentence.split()
['Hello,', 'world.']
≫ from nltk.tokenize import word_tokenize
≫ tokens = word_tokenize(sentence)
≫ tokens = [el for el in tokens if el.isalnum()]
['Hello', 'world']
```

Homogenization

- The goal is collapse all equivalent words (i.e., semantically equivalent) in a unique representative
  - Different forms of a word:
    organize, organizes, organizing, ... $\rightarrow$ organize
  - Derivationally related words with similar meanings:
    democracy, democratic, democratization, ... $\rightarrow$ democracy
- Stemming vs lemmatization
  - **Stemming** chops of the ends of words using a list of suffixes
  - **Lemmatization** usually refers to doing things properly with a vocabulary and morphological analysis of words, aiming to return the base or dictionary form (lemma) of a word.
  - A practical difference is that lemmatizers output complete words.
- If using case sensitive implementations, we need to lowercase the tokens as a preliminary step

Stemming

```
>> import nltk.stem
>> s = nltk.stem.SnowballStemmer('english')
>> s.stem('image') → 'imag'
>> s.stem('images') → 'imag'
>> s.stem('organize') → 'organ'
>> s.stem('organizing') → 'organ'
>> s.stem('organ') → 'organ'
```

Lemmatization

- NLTK recurs to WordNet, a lexical database for the English language. Wordnet groups English words into sets of synonyms called synsets.
- With contextual information (the grammatical role of the word) `lemmatize()` can filter grammatical differences

```
>> from nltk.stem import WordNetLemmatizer
>> wnl = WordNetLemmatizer()
>> wnl.lemmatize('image') → 'image'
>> wnl.lemmatize('images') → 'image'
>> wnl.lemmatize('organize') → 'organize'
>> wnl.lemmatize('organizing') → 'organizing'
>> wnl.lemmatize('organizing', pos='v') → 'organize'
>> wnl.lemmatize('organ') → 'organ'
```

Other homogeneization tasks

Part of Speech Tagging

- Part-of-speech (POS) tagging is the process of assigning a word to its grammatical category (noun, verb, adverb,), in order to understand its role within the sentence.
- POS tagging is provides the contextual information to `lemmatize()` to filter grammatical differences.

```
≫ from nltk import pos_tag
≫ tokens = word_tokenize('This is a simple sentence')
≫ pos_tag(tokens)
[('This', 'DT'), ('is', 'VBZ'), ('a', 'DT'), ('simple', 'JJ'),
('sentence', 'NN')]
```

N-gram detection

- Identification of groups of words that occasionally appear together, and have an inherent semantic value
- E.g.: Machine learning, big data, etc

Cleaning

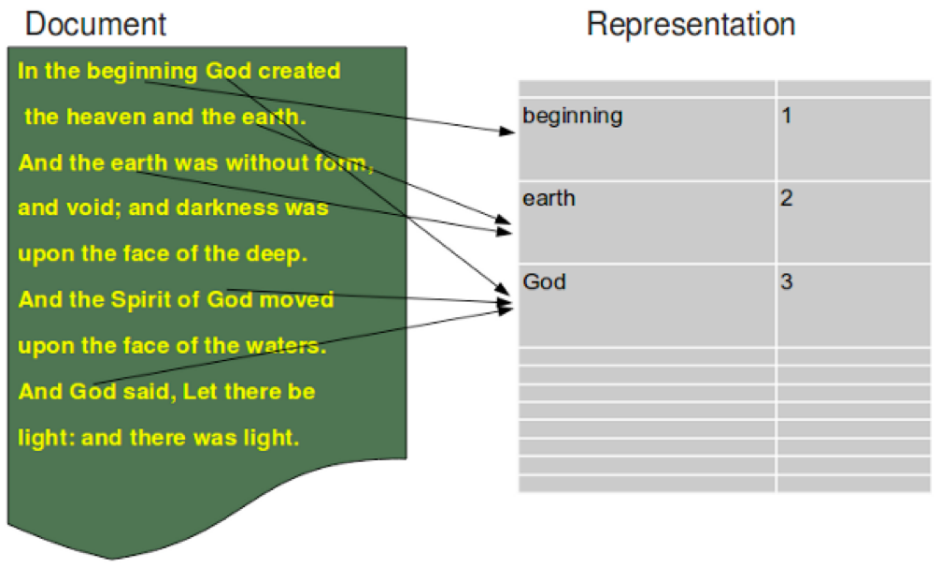
The goal is to remove irrelevant words

- Stopwords: Words that appear very often in all sorts of different contexts. They are removed by using stopwords lists
- Very rare terms: We may also want to remove words that appear in a very reduced number of documents in the collection
- Very common terms: We may also want to remove words that, in spite of not being stopwords, are very common for a particular dataset, or simply do not have significant semantic value for the application at hand

Vectorization: Bag-of-words representation

The goal is to convert the sequence of homogenized and cleaned tokens into a numeric vector

- Bag-of-words representation: counts how many appearances of each word occur in each document
- Each position in the vector is associated to a unique word in the vocabulary, so vectors are typically very sparse



Term frequency - Inverse document frequency (TF-IDF)

We want a high value for a given term in a given doc if that term occurs often in that particular doc and very rarely anywhere else

- $\mathrm{TF}(w, d) = \frac{\mathrm{BoW}(w,d)}{\#\text{ words in } d}$
- $\mathrm{IDF}(w) = \log \frac{\#\text{ docs}}{\#\text{ docs with term } w}$
- $\mathrm{TF-IDF}(w, d) = \mathrm{TF}(w, d) \times \mathrm{IDF}(w)$

The IDF term emphasizes the words that appear in a reduced number of documents.

Exercises

Exercise

- Preprocess the text data in the NSF dataset, completing all the steps that have been reviewed in the presentation
- Follow Chapter 2 of the provided 'DTU02901.ipynb' notebook. As a result, the following variables will be available:
  - `D`: A gensim dictionary. Term strings can be accessed using the numeric identifiers. For instance, `D[0]` contains the string corresponding to the first position in the BoW representation.
  - `corpus_bow`: BoW corpus. A list containing an entry per project in the dataset, and consisting of the (sparse) BoW representation for the abstract of that project.
  - `NSF_data`: A list containing an entry per project in the dataset, and consisting of metadata for the projects in the dataset

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

Topic modeling tools

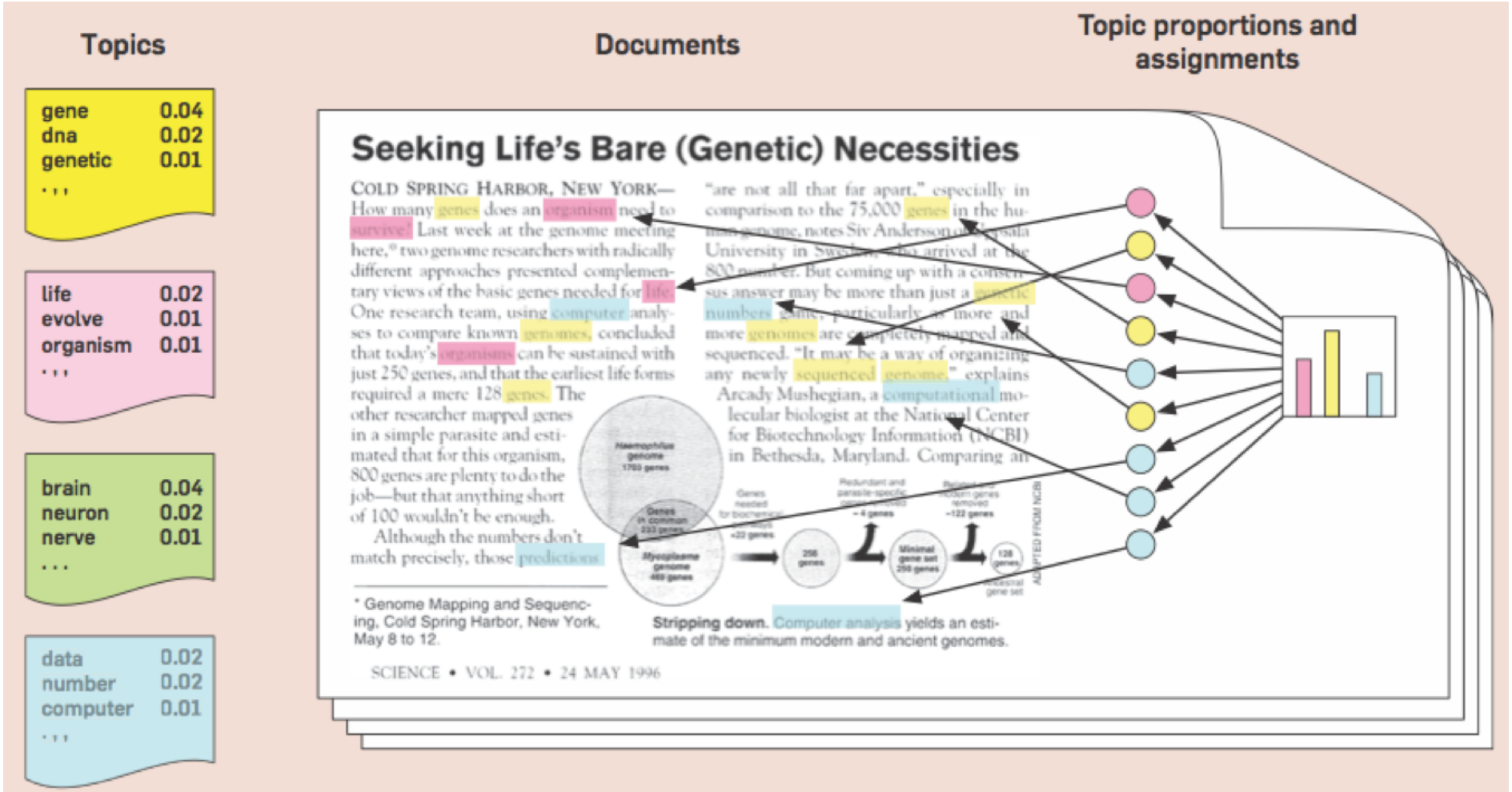
- **Gensim** is an NLP toolbox Developed by Radim Rehurek, and specialized in topic modeling and semantic analysis of texts. It includes:
  - Tools for BoW codification of documents
  - Tools for TF-IDF representation of documents
  - Latent Semantic Indexing (LSA/LSI)
  - **Latent Dirichlet Allocation (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`

Latent Dirichlet Allocation

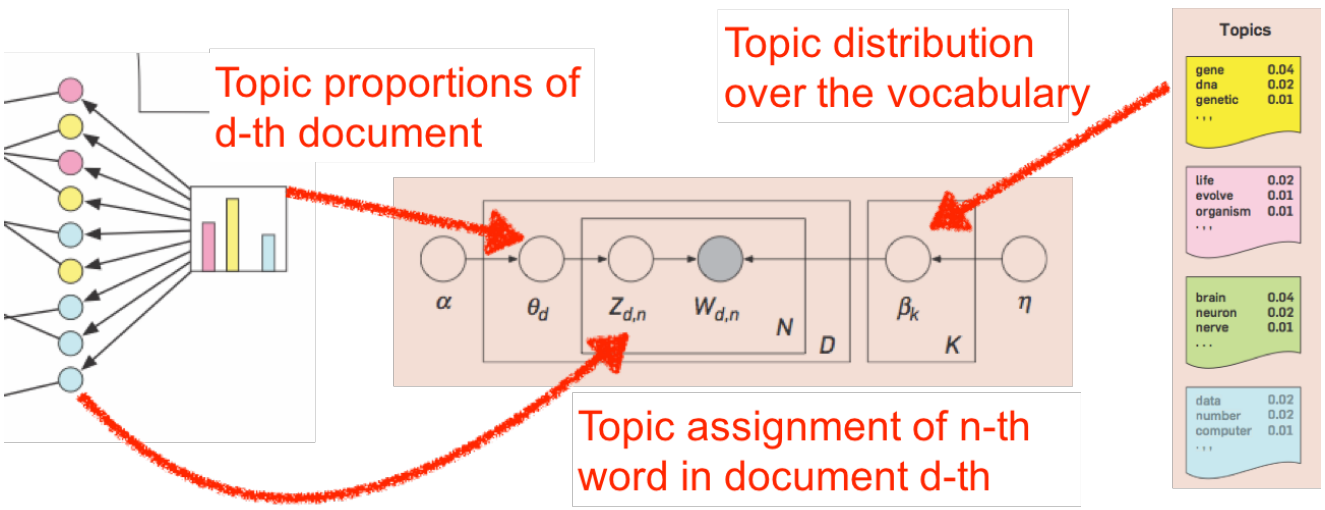


(Image taken from
http://www.cs.columbia.edu/~blei/papers/Blei2012.pdf)

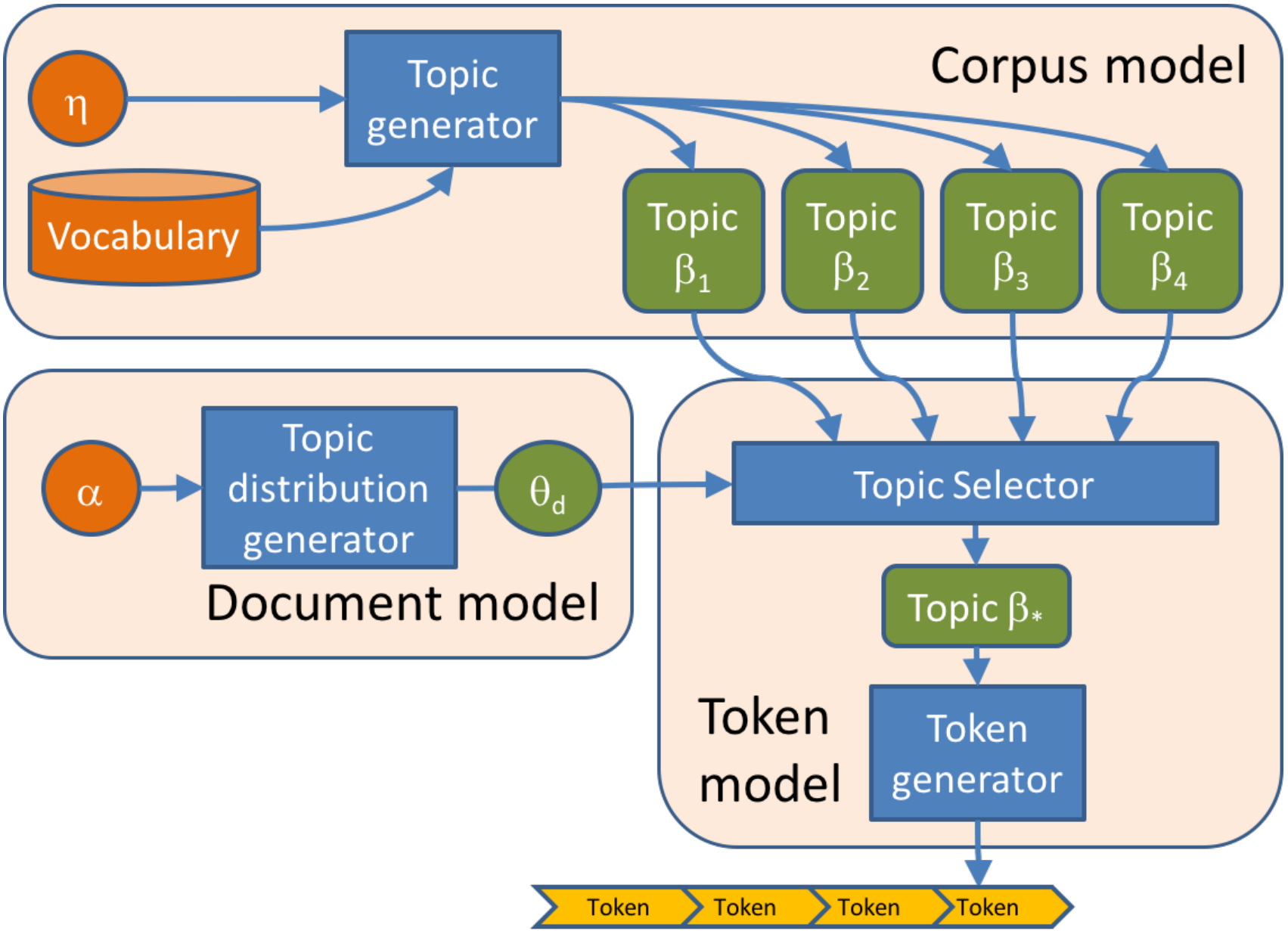
Understanding LDA

LDA is a generative probabilistic model:

- It assumes that documents have been generated according to some probability model with some unknown parameters an certain hidden variables
- The corpus data is used to infer the topic structure (hidden variables) from the words of documents (observed variables)



LDA generative model



LDA generative model (II)

- Topic generator:
  - Generates tokens distributions by means of a Dirichlet distribution with concentration parameter η.
  - Each topic is therefore characterized by a vector $\boldsymbol{\beta}_t$ that indicates the term distribution for the given topic
  - Large η 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 α.
  - Large α implies that for most documents only a few topics will be relevant.
  - For each word in the document:
    - Select a topic according to the topic distribution for the document
    - Select a word from the word distribution for the selected topic

LDA generative model (III)

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

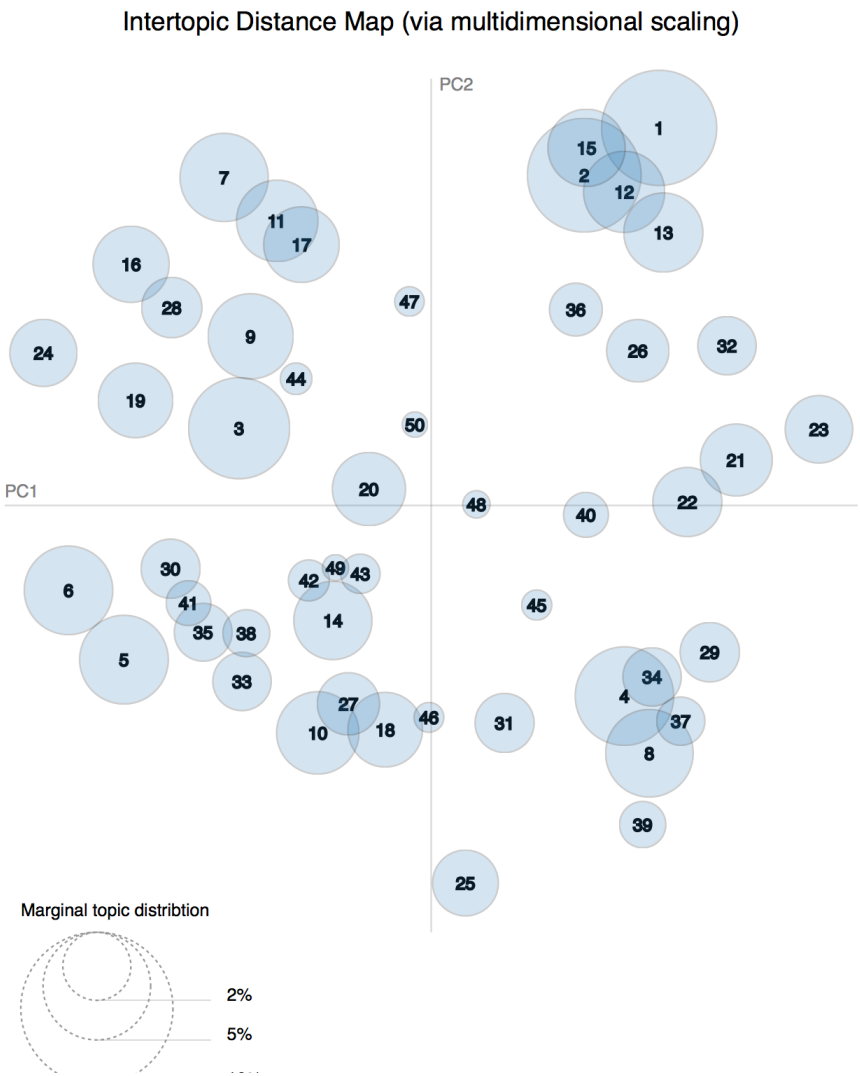
$$p(\boldsymbol{\beta}_{1:K}, \boldsymbol{\theta}_{1:D}, \mathbf{z}_{1:D}, w_{1:D}) = \prod_{i=1}^{K} p(\boldsymbol{\beta}_i) \prod_{d=1}^{D} p(\boldsymbol{\theta}_d) \left(\prod_{n=1}^{N_d} p(z_{d,n}|\boldsymbol{\theta}_d) p(w_{d,n}|z_{d,n}, \boldsymbol{\beta}_{1:K}) \right)$$

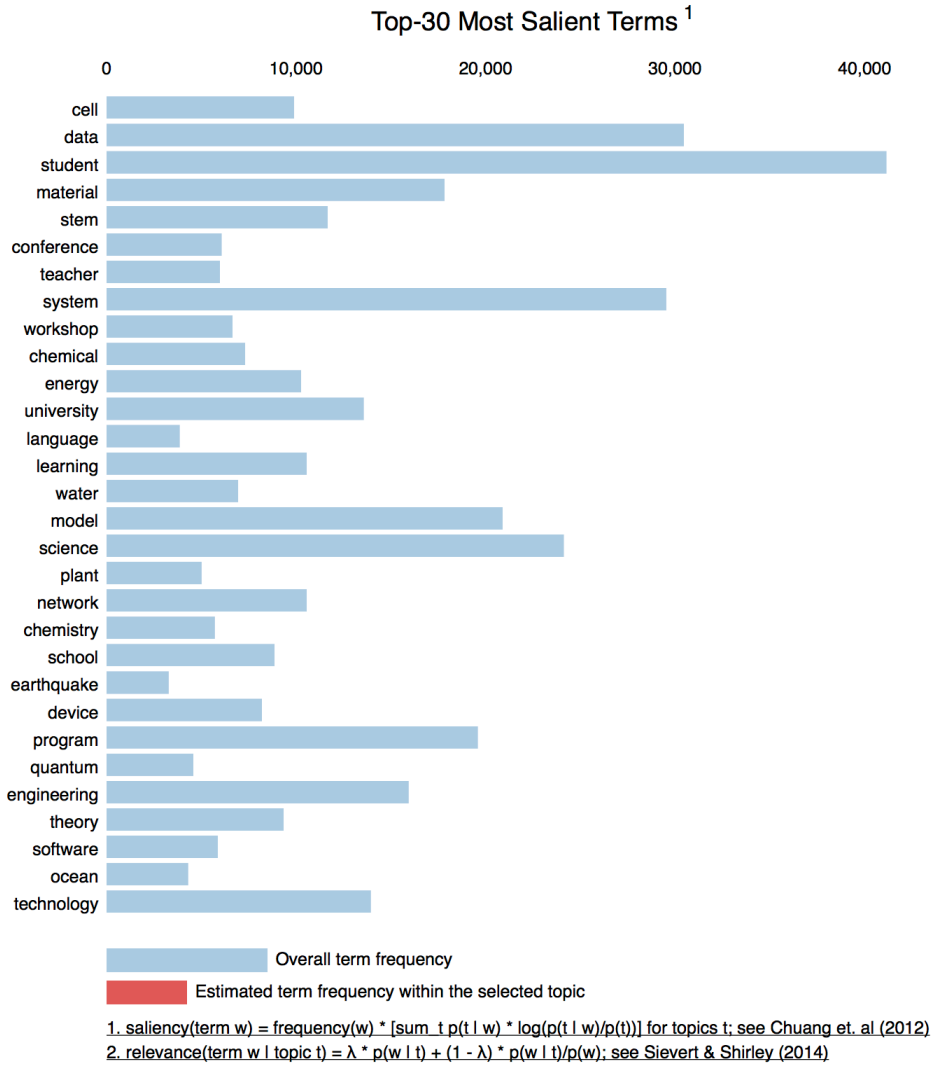
- Goal: computing the conditional distribution of the topic structure given the observed documents

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

- 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, $\mathbf{z}_{1:D}$)

Topic visualization using pyLDAvis





Some open issues in topic modeling

- Joint use of Topic Models and Word2Vec
  - Word2Vec provides a semantically inspired vector representation of words
  - Using this representation space, models can be aware of statistical dependencies among words
  - Word2Vec relies on Deep Learning. The representation is given by the hidden layer of a network trained to predict words context.
- Matching the topics of models trained with heterogeneous vocabularies
- Interactive topic models incorporating domain-expert knowledge

Exercises

- Complete notebook subsections 3.1. and 3.2., and study how Topic Models can be trained and visualized using Gensim and pyLDAvis modules.
- Complete notebook subsection 3.4. to know about some useful Gensim functions to work with the topic model
- Complete the exercises in Subsection 3.4.
  - Build a function that returns the most relevant projects for a given topic
  - Build a function that computes the semantic distance between any two selected projects.
  - Compute the distance between institutions as the median of all distances among all projects of both institutions

Contents

Graph Analysis

- We are in a position to build graphs to visualize and explore relationships among projects and institutions
- Project graphs:
  - Each project is a network node, and the metadata is already available in the constructed dataset.
  - The weight of the link between any two nodes can be computed from their JS divergence as

$$exp(-\gamma \cdot \text{divergence})$$

  - It is convenient to impose a threshold value to obtain a sparser graph.
- Institution graphs:
  - Each institution is a network node, and the metadata is already available in the constructed dataset.
  - The weight of the link between any two nodes can be computed as the mean (or median value) of

$$exp(-\gamma \cdot \text{divergence})$$

for any pair of projects from the two institutions

Assignment

- Generate a graph of projects and/or institutions using the topic model you build during the session
- If necessary, you can sample the nodes
- Visualize the graph using Matlab tools of the Gephi software, and run any community detection algorithm to group projects or institutions that are semantically similar