# Recommender system for mentor-mentee matching

Baptiste Raemy PMI interview, July 2017

#### Definition

 Goal: match mentor and mentee considering their preferences and attributes

 Input data: DBLP Computer Science Bibliography dataset: list of authors and publications

## Recommender System

Based on the solution of the Netflix Price

M = Mentor m = mentee t = topics

	M1	M2	МЗ	M4
m1	1		3	2
m2	3	2	4	
m3		5		4
m4	1		0	2

	t1	t2
m1	1	2
m2	0	4
m3	2	0
m4	1	3

	M1	M2	МЗ	M4
T1	1	3	2	5
T2	3	3	4	1

#### How to build the matrix?

- Mentee Matrix:
  - the mentee give a rate for each of the given topics
  - Better solution: the mentee answer questions and each question gives some point for each topics
- Mentor Matrix: use the Dblp files to extract the main topics among all the publication and determine the fields of each mentor

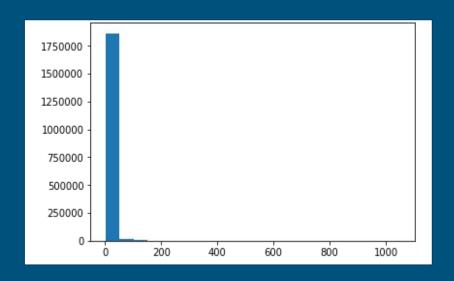
#### Dblp

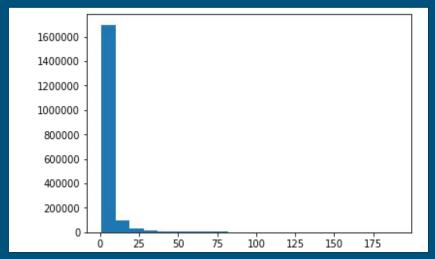
- Xml file of ~50 million of line (2GB)
- Contains article, book, master thesis,...
- Give the author of each publication
- Contains list of possible authors
- Homonymous are distinguished by concatenating an id
- There are disambiguation pages that contains publication of homonymous but they are not distinguished yet

## Preprocessing

- Build a mapping author->title
- Group by author.
- Same author can have different name:
  - John Doe or J. Doe
  - Use the item about person that will map John Doe and J. Doe to the same author ID
- Remove the disambiguation pages

### Remove Disambiguation





Authors with more than 1010: 2 (0.0001%)
Authors with more than 500: 149 (0.0079%)
Authors with more than 300: 698 (0.0369%)
Authors with more than 200: 2002 (0.1059%)
Authors with more than 190: 2249 (0.1189%)
Authors with more than 50: 26952 (1.4254%)

#### **Topics Extraction**

- Preprocessing:
  - Tokenization, lower case
  - Stopword filtering
  - stemming (prefered over lemmatizer):
    - stem(Algorithmic, algorithm) => (algorithm,algorithm)
    - lem(Algorithmic, algorithm) => (Algorithmic, algorithm)

## Latent Dirichlet allocation (LDA)

- Each documents represent a mixtures of topics: a documents contains words that belong to different categories.
- The goal of LDA is to retrieve those sets of words used to create the documents by using probability
- Work better with long document that short document

We will use the implementation of Gensim for LDA and use the NLTK library for the preprocessing

#### LDA

#### Assumption:

- A author is characterized by a subset of topics => his publications are a mixture of topics => group the titles together to have one document per authors
- We set the number of topics to 20. To many topics will create intersection among them

# Measuring the level of expertise of a mentor

- 2 solutions:
  - LDA output a score of the document (titles concatenated) for each author
  - LDA output a score for each titles for each author and then add all the score
    - add is better than average in that case

 Scale all the score from 0 to 10. For each topics, the mentor with the highest score get 10, the worst 0 and then scale all the others

#### Recommendation method

Matrix multiplication: Mentee X Mentor = score matrix

#### Matching:

- Need to make some choice:
  - 1-1 mapping (1 mentee, 1 mentor)
  - N-1 mapping (N mentee, 1 mentor)
  - o ..
  - One pair with good score, the second with bad score
  - 2 pairs with average score
- Use a matching algorithm: Gale-Shapley algorithm for example
  - Each mentee ask his highest match, mentor answer maybe if alone or if the score of such mentee is better)

## Evaluation of the system

First thing to do is to test edge case:

 Mentee only like one topics => mapped with the with the highest expertise in that topics