Clustering and Creating Recommendations with Scholarly Publication Data

Mustafa Can Çavdar & Tolga Yılmaz

Problem Description

- Scholarly websites provide list of papers
  - Google Scholar, CiteUlike, Citeseerx, dblp
- Finding a particular paper is easy if it is known what is exactly needed.
- Related articles and related authors to a topic cannot be found easily.

Motivation

- While researching on the web, we want to find the people who are the most related ones to the topic.
- People want to see who produce articles from a certain field and who have similar works.
- It is not seen that people can see similar papers with stating a certain paper.
- It is a big amount of data to play with and it is a good opportunity to create an open source implementation.

Functionalities

- Basic Search
  - Article search
  - Author search
- Given an article, find similar articles.
- Given a field, find related articles.
- Given an author, find similar authors.

Methodology

- Collect Data.
- Search
- Find similar entities
- List of tools
  - oaiharvester, Lucene, MySQL, jQuery/javascript, Java servlets, Tomcat, Bootstrap ui.

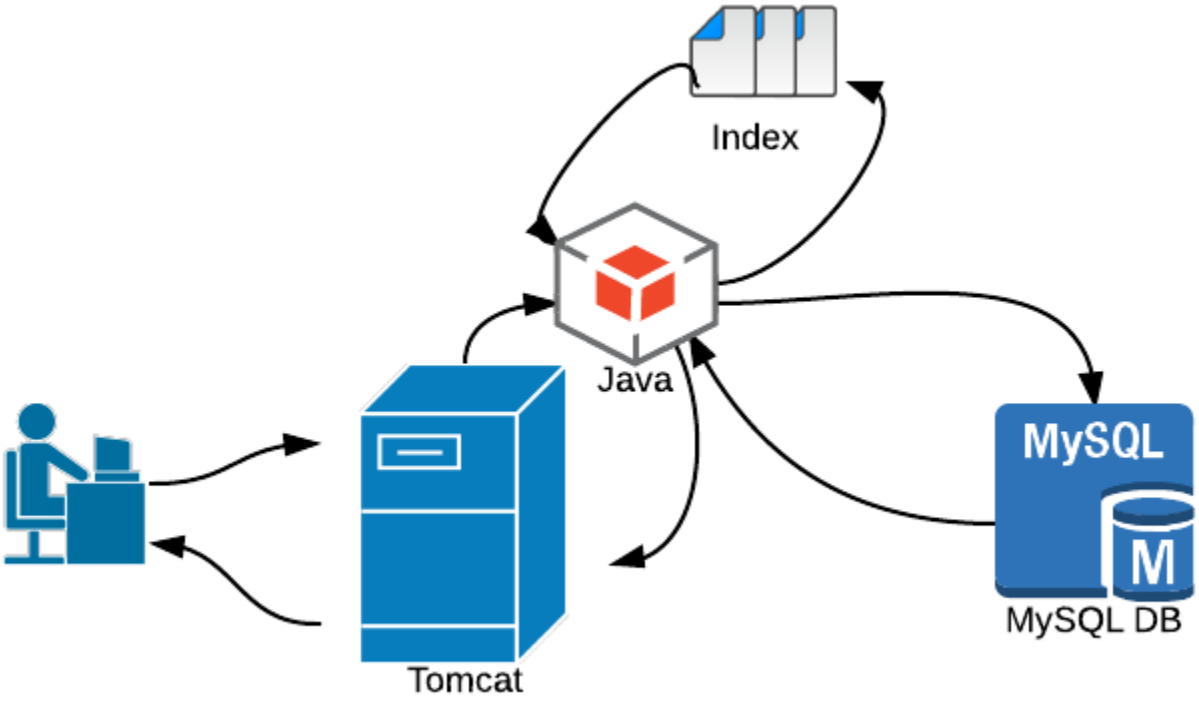
Collecting Data and parsing

- ***CiteUlike*** gives a snapshot of daily article postings. Given an id from this data, you need to crawl their website -> Banned.
- ***Google Scholar*** does not provide the data but one crawl their website -> Banned.
- ***Dblp*** gives their entire dataset. But the set does not include abstracts of articles. Not good for similarity calculations.
- ***Citeseerx*** makes their data set through an OAI interface for querying.
  - We collected the data set using an OAI harvester. The size is ~ 943MB
- This file is a corrupted XML file which needs to be parsed. After parsing this we have a file for each article. We have limited this number to 250000 for performance issues. Total size here is ~201 MB (976 MB on disk).

Search

- For this part, we built a basic search engine on top of Lucene which is a Java based, open source search engine library.
- Indexing
  - is done using existing Lucene code.
  - The constructed index is around 67 MB.
- Searching
  - is implemented on top of Lucene.
- In order to retrieve the resulting documents of the search process, we have also inserted them into a MySQL database.

Search Flow

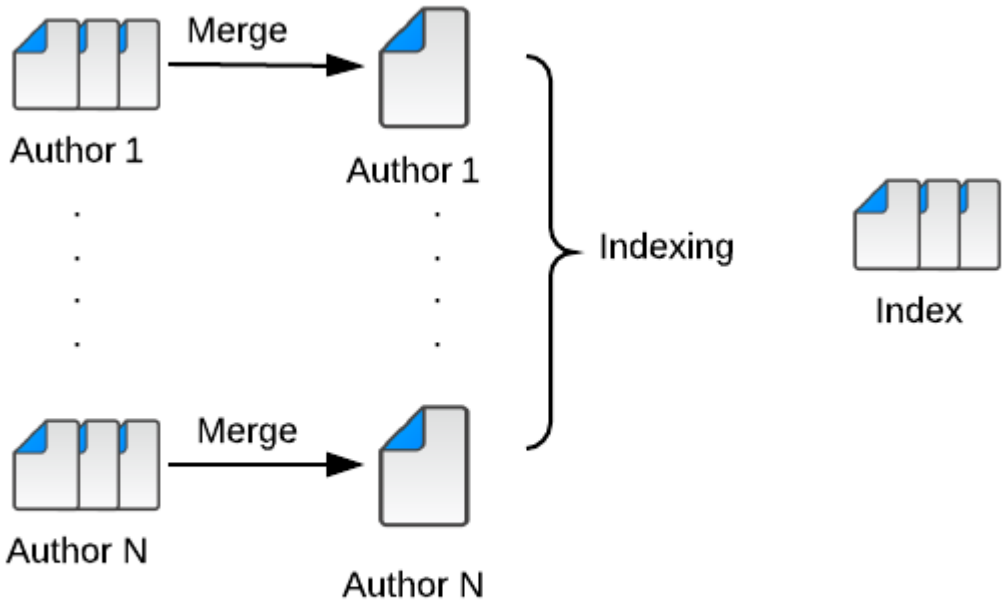


Given an article, find similar articles

- The article is seen as a query. Finding similar articles then is a simple search process.

Given an author, find similar authors

- The article is seen as a query. Finding similar articles then is also a search process over the built index. (Note that this index is a seperate one.)



Experimental Results

- We have an efficient search mechanism.
- Evaluation of effectiveness of results is possible
  - Over a small set of queries, find P@10 and MAP which are very high at first glance.
  - Other methods are not possible due to lacking comparison source.

THANK YOU FOR LISTENING!

Any Questions?