CS387 Project: Final Report

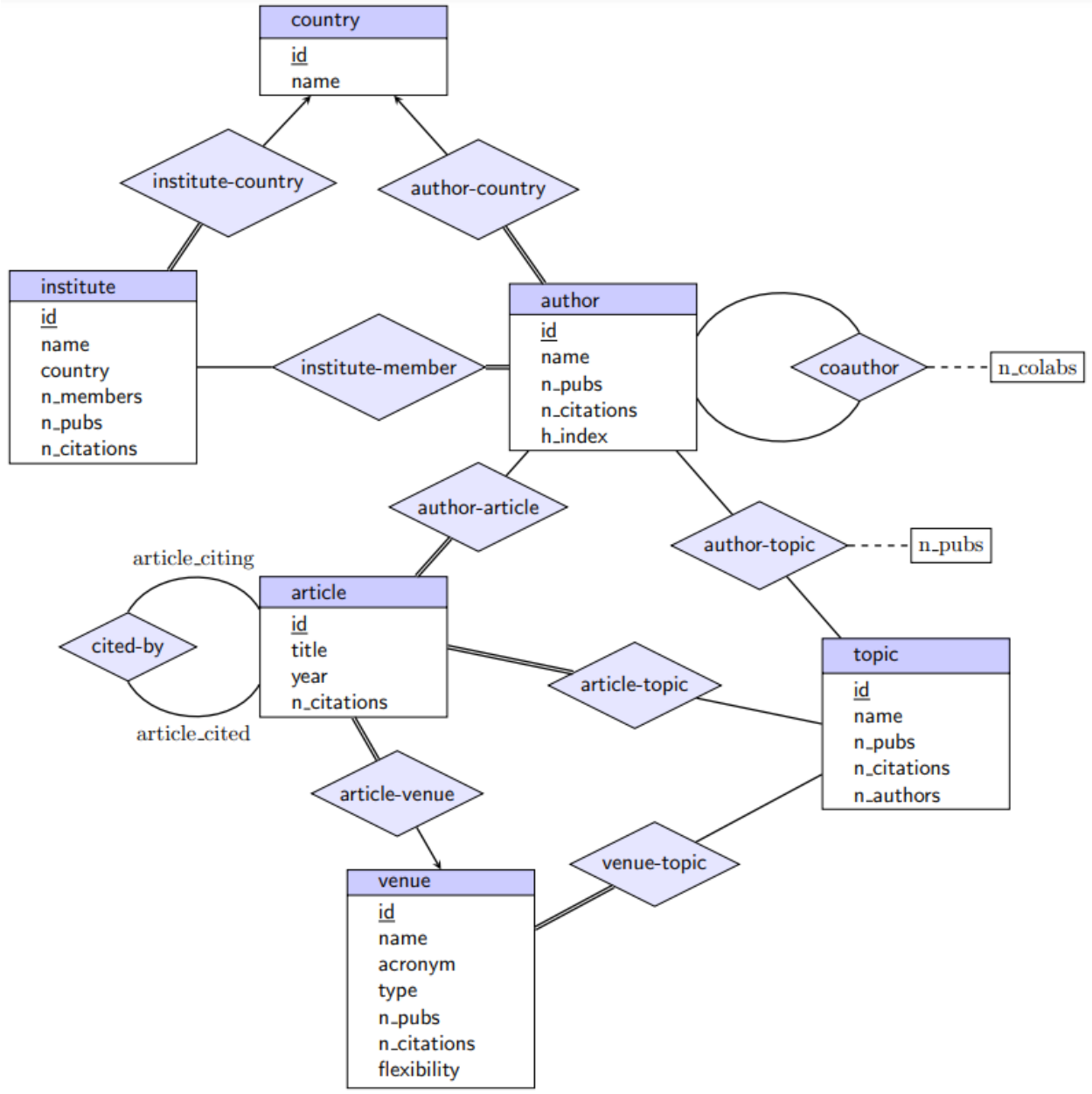
Github Link

https://github.com/CS387-Spring2022-CourseProject/CStem

Team

Name	Roll Number
Adarsh Kumar	19D180003
Adithya Bhaskar	190050005
Devansh Jain	190100044
Harshit Varma	190100055

ER Diagram


country
id
name
institute-country
author-country
institute
id
name
country
n_members
n_pubs
n_citations
institute-member
author
id
name
n_pubs
n_citations
h_index
coauthor
n_colabs
author-article
author-topic
n_pubs
article_citing
article
id
title
year
n_citations
cited-by
article_cited
article-topic
topic
id
name
n_pubs
n_citations
n_authors
article-venue
venue-topic
venue
id
name
acronym
type
n_pubs
n_citations
flexibility

Normalized Logical Schema

This schema is used for the CSV files, which are in turn used for loading the data into the Neo4j database. Some relations (like author_country, article_venue) can be reduced to attributes, but are kept as relations since queries are more efficient that way in Neo4j (reference: https://neo4j.com/blog/dark-side-neo4j-worst-practices/)

Integrity Constraints

- Primary and Foreign key constraints are as per the schema given above
- Other constraints are given by the ER diagram
- No attribute is allowed to be null in any table

Materialized Views

Materialized views are not needed.

DB Creation and Inserting Data

Prior to loading the data in the Neo4j DB, we store it in CSV files according to the schema. Then a python script (similar to the one in Lab-6) is used to create cypher queries to insert the data into the Neo4j DB. However, this is too slow for large datasets. Thus, for large datasets we use py2neo's bulk loading interface which is much faster.

User Interface / Requirements

- `(/home)`
  Contains tabs for `/institutes, /authors, /venues, /topics, /articles`

- `(/institutes)`
  Shows a table with columns corresponding to the institute entity's attributes
  Institute names are clickable and redirect to `/institutes/<institute_id>`
  Rows are sorted according to a default metric (citations per member)
  User is able to:
  - sort by any of the defined metrics (dropdown)
  - filter by the country (dropdown)
  - filter by topics (dropdown): only articles having these topics will be used for re-computing the metrics
  - search by name (text input)
  - specify a time range (a slider with 2 movable ends): metrics will be re-computed after this using only the articles which were published in this time range

- `(/institutes/<institute_id>)`
  Show the institute details (name, country, metrics)
  Show a table with all the members of the institute, member names redirect to `/authors/<author_id>`
  Charts:
  - bar graph: #publications vs year
  - bar graph: #citations vs year

- `(/authors)`
  Shows a table with columns corresponding to the author entity's attributes
  Author names are clickable and will redirect to `/authors/<author_id>`
  Rows are sorted according to a default metric (total citations)
  User is able to:
  - sort by any of the defined metrics (dropdown)
  - filter by the country (dropdown)
  - search by name (text input)
  - filter by topics (dropdown): only articles having these these topics will be used for re-computing the metrics

    - specify a time range (a slider with 2 movable ends), metrics will be re-computed after this using only the articles which were published in this time range

- `(/authors/<author_id>)`
  Show author's details (name, country, institutes), metrics, top-5 topics, and a table containing publications (name, year, citations) authored by the author.
  Charts:
    - bar graph: publications vs year
    - bar graph: citations vs year
    - pie chart: number of publications vs topics (top-5 only)
    - co-authorship graph for the author (nodes are authors, weighted undirected edge if they've collaborated, with weight = no. of co-authored articles)

- `(/articles)`
  Shows a table with columns corresponding to the article entity's attributes
  Article titles are clickable and will redirect to `/articles/<article_id>`
  Rows are sorted according to a default metric (total citations)
  User is able to:
    - sort by any of the defined metrics (dropdown)
    - search by name (text input)
    - filter by venues (dropdown)
    - specify a time range (a slider with 2 movable ends), metrics will be re-computed after this using only the articles which were published in this time range

- `(/articles/<article_id>)`
  Shows article entity's attributes along with the article's venue, topics and metrics.
  Charts:
    - bar graph: #citations vs year
    - citation graph for the article

- `(/venues)`
  Shows a table with columns corresponding to the venue entity's attributes and the list of topics for that venue
  Rows are sorted according to a default metric (average citations)
  Venue names are clickable and will redirect to `/venues/<venue_id>`
  User is able to:
    - sort by any of the defined metrics (dropdown)
    - filter by the type (dropdown)
    - filter by topics (dropdown)
    - search by name (text input)

- `(/venues/<venue_id>)`
  Show venue details (name, acronym, type, topics, metrics)
  Charts:

    - bar graph: #publications vs year

- `(/topics)`
  Shows table with topic names and their metrics
  Rows are sorted according to a default metric (average citations)
  Topic names are clickable and will redirect to `/topics/<topic_id>`
  User is able to:
    - sort by any of the defined metrics (dropdown)
    - search by name

- `(/topics/<topic_id>)`
  Shows topic name and metrics
  Charts (can be used to visualize a topic's popularity over time):
    - bar graph: #publications with this topic vs year
    - bar graph: #citations of publications with this topic vs year

Backend Description

Due to the highly dynamic nature of metrics that were to be displayed on the frontend which would change with the time ranges and selections specified, much of the computation had to be done on-the-fly. The graph processing frameworks Neo4j and Spark were used towards this end. The latter was incorporated into python scripts via the pyspark library, and the former's use was facilitated by the inclusion of the FastAPI framework.

For instance, the frontend has the provision of a 2-way range slider which can be used to set the 'reference time frame' of the entities, and attributes like the number of citations and publications have to be recomputed based on the range of values selected by the user. On top of this, we also implemented filtering of rows based on topics/types of venues/countries. Instead of manipulating the result on the frontend, we aim to fetch the filtered data directly from the backend.

Queries like the citation/co-authorship graph have a depth parameter associated, which dynamically adjusts the upper bound on the path length, thus requiring no pruning after the fetch. Note that the extra queries at the backend serve as the tradeoff against redundancy in the ER layout.

Frontend Description

The aim of the frontend layout was to crisply but concisely convey all the information that the user required via the use of quality-of-life improvements wherever possible. Data has been organized into tabular or graphical formats wherever possible, with pagination and attribute-wise sorting being incorporated into all tables.
Most tables also have a search bar that lets the user filter rows based on whether the input string matches any of its corresponding columns. We have also provided the ability to restrict the results to, e.g., specific topics. In some cases (such as for Institutes), this mandates a complete recomputation of the displayed metrics which is seamlessly and transparently offloaded to the backend via the use of a common query service.
The attributes of these tables are also designed to update accordingly based on the filters. The manipulations involved are handled via the query service with proper two-end interactions.

Transaction Descriptions and Queries

Cypher

Cypher queries are used for loading all data required for tables, charts, graphs, and recomputing metrics at the frontend.

Spark

Spark is used to compute the various metrics and fill up some of the CSV files after extracting the base data.

- n_citations for articles
- n_citations, n_pubs, h_index for authors
- n_pubs, n_members, n_citations for institutes
- n_articles, n_authors, n_citations for topics
- n_pubs, n_citations, flexibility for venues
- n_colab for coauthor
- Topics an author publishes in with the number of publications per topic, extracted from the author's articles (this is used to fill up article_topic.csv)

Indices for Optimization

All our queries require operations that use id's. Thus, we added indices over each node's id. We also have an index over article.year as it is involved in range queries.
Reference: https://neo4j.com/docs/cypher-manual/current/indexes-for-search-performance/

Technology

- Neo4j database
- FastAPI backend
- Angular frontend
- vis-network for rendering graphs
- ApexCharts for charts
- PySpark for initial metrics recomputation
- py2neo for bulk loading the data

Data Generation

This was an unexpectedly challenging portion of the project.
We tried and tested three different raw data:

1. dblp.xml (https://dblp.org/faq/1474679.html)
   79457393 nodes in XML tree
   6061206 Total Publications (3010090 Formal Conference Papers)
   2977403 Authors and 7813 Venues
2. DBLP-Citation-network V13 (https://www.aminer.org/citation)
   5,354,309 Papers and 48,227,950 citation relationships
   Dataset was last updated on 2021-05-14
3. ArnetMiner (https://www.aminer.org/aminernetwork)
   2,092,356 papers and 8,024,869 citations relationships
   1,712,433 authors and 4,258,615 collaboration relationships

We performed two tests on CStem:

1. Functional testing - we artificially generated a small citation graph
   14 papers and 23 citation relationships
   8 authors and 4 institutions
   4 venues and 3 topics

   We tested for the correctness of data displayed on different pages.
   We also verified for corner cases with authors with no papers, institutions with no authors, papers with no references, and papers with no citations.

2. Load testing - we extracted data from the ArnetMiner dataset
   12783 papers and 19208 citation relationships
   16039 authors and 405 institutions
   9 venues and 3 topics

   We chose ArnetMiner because of the availability of Author-Paper id mapping (`AMiner-Author2Paper.txt`) which was absent in other datasets. The major issue was caused due to multiple aliases of names used by the same author in different conferences.

   One major issue faced across all three datasets was inconsistency in author to institution mapping. So, we have randomly assigned institutes and countries to the author from a list of well-known institutions from around the world.

   a. `institutes.py`
      In: `countries.csv` (code, latitude, longitude, name)
      In: `institutions.csv` (name, countryabbr)
      Out: `country.csv` (id, name)

Out: `institute.csv` (id, name)
Out: `institute_country.csv` (institute_id, country_id)

b. `article.py`
In: `AMiner-Paper.txt`
In: `venue_topic.csv` (venue_id, topic_id)
Out: `article.csv` (id, title, year, venue_id)
Out: `article_topic.csv` (article_id, topic_id)

c. `citation.py`
In: `AMiner-Paper.txt`
In: `article.csv` (id, title, year, venue_id)
Out: `cited_by.csv` (article_id_1, article_id_2)

d. `author_article.py`
In: `AMiner-Author2Paper.txt`
In: `article.csv` (id, title, year, venue_id)
Out: `author_article.csv`(author_id, article_id)
Out: `author_id.csv` (author_id)

e. `author.py`
In: `AMiner-Author.txt`
In: `institute_country.csv` (institute_id, country_id)
In: `author_id.csv` (author_id)
Out: `author.csv` (id, name)
Out: `author_country.csv` (author_id, country_id)
Out: `institute_member.csv` (institute_id, author_id)

The generated csv files are then processed using spark to compute the metrics.

Test Results

Functional Testing

We designed synthetic data for functional testing as described earlier:

- Testing whether the metrics/analytics computed/recomputed were correct. On a small data size, this was manually verifiable.
- Testing whether the metrics/analytics computed/recomputed were consistent everywhere. For example, the sum of n_citations vs. years should equal the total citations.
- Testing on edge cases, e.g., an institute with no author and an author with no articles.
- Testing various forms with valid/invalid inputs: most forms contained selections of some sort, so invalid inputs were not an issue.

Load Testing

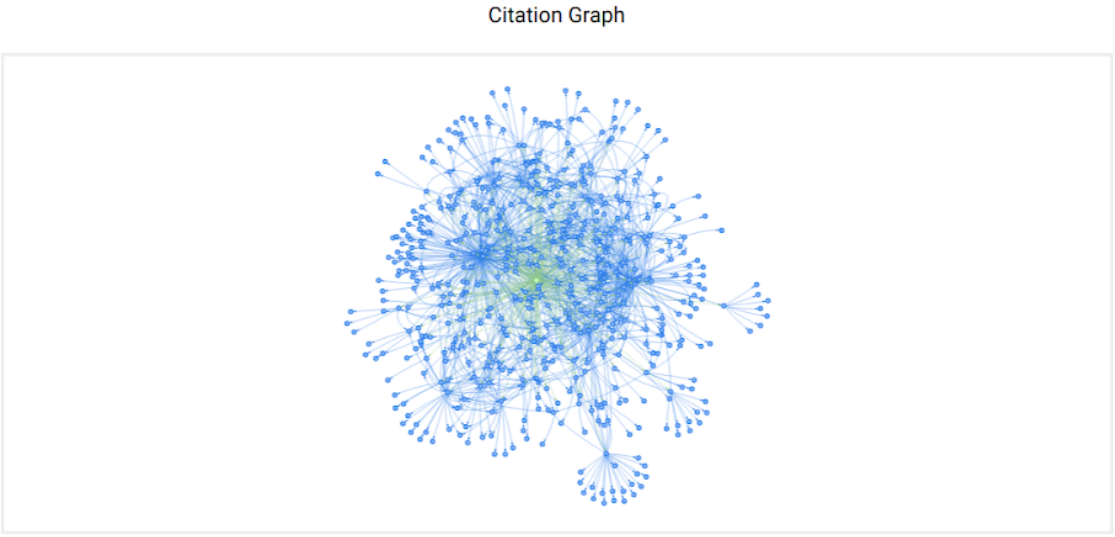
We used real data for load testing:

- 29,401 nodes and 146,021 relationships
    - 12,733 articles
    - 16,007 authors
    - 19,208 citation relationships
    - 404 institutes (real data for this wasn't available, had to assign randomly)
    - 9 venues
    - 3 topics
- The response time of queries:
    - /institutes: < 1s
        - filtering, sorting: < 1s
        - time range queries: 1s
    - /institute/<id>: < 1s
    - /authors: 2s (pagination at the backend itself can reduce this)
        - sorting: < 1s
        - filtering: 5s (pagination at the backend, better query can reduce this)
    - /author/<id>: < 1s
    - /articles: 3s (pagination at the backend itself can reduce this)
        - filtering: <1s
        - sorting: < 1s
        - searching: < 1s
    - /article/<id>: < 1s (for citations less than 50)
    - /topics: < 1s
    - /topic/<id>: < 1s
    - /venues: < 1s
    - /venue/<id>: < 1s
- The rendering of large scale graphs: when graphs were too large, the animation engine of vis-network seemed to break and kept "jiggling" some nodes and edges. Rendering of

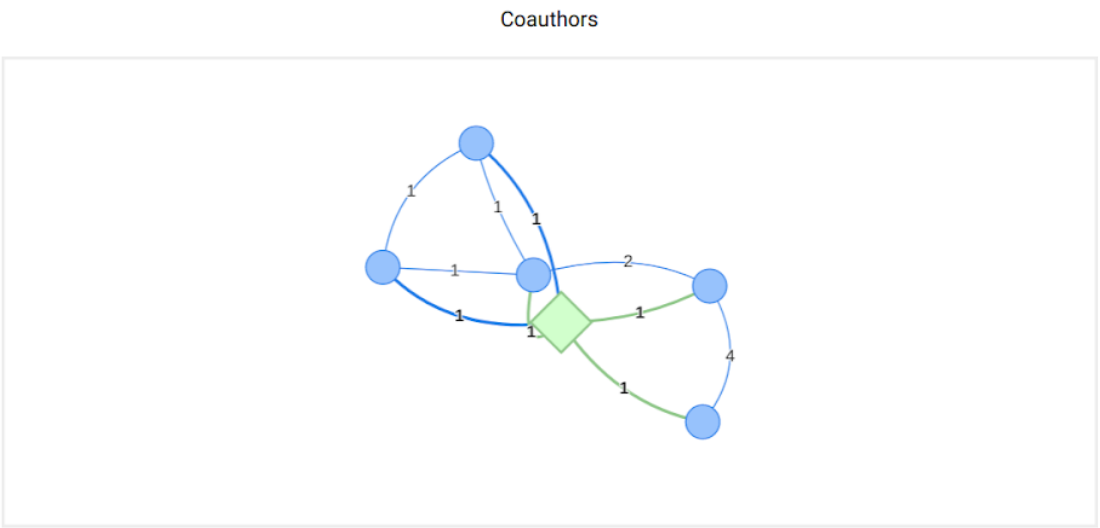
these graphs also took longer, especially when the depth parameter for citation graph queries was high.

Key Results (on real data)

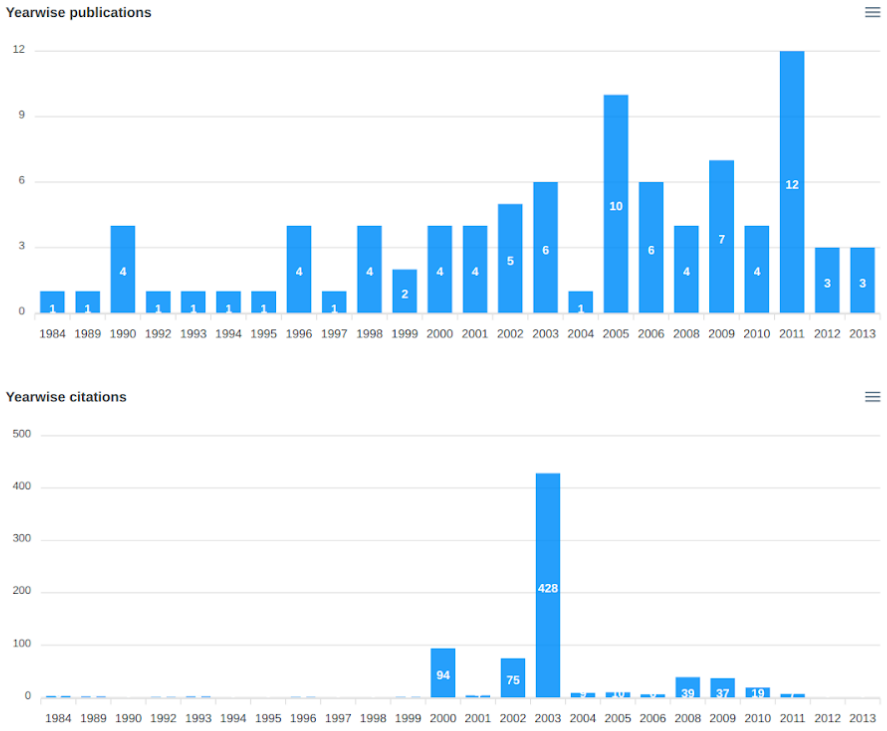
- Large Citation Graphs (depth set to 2)



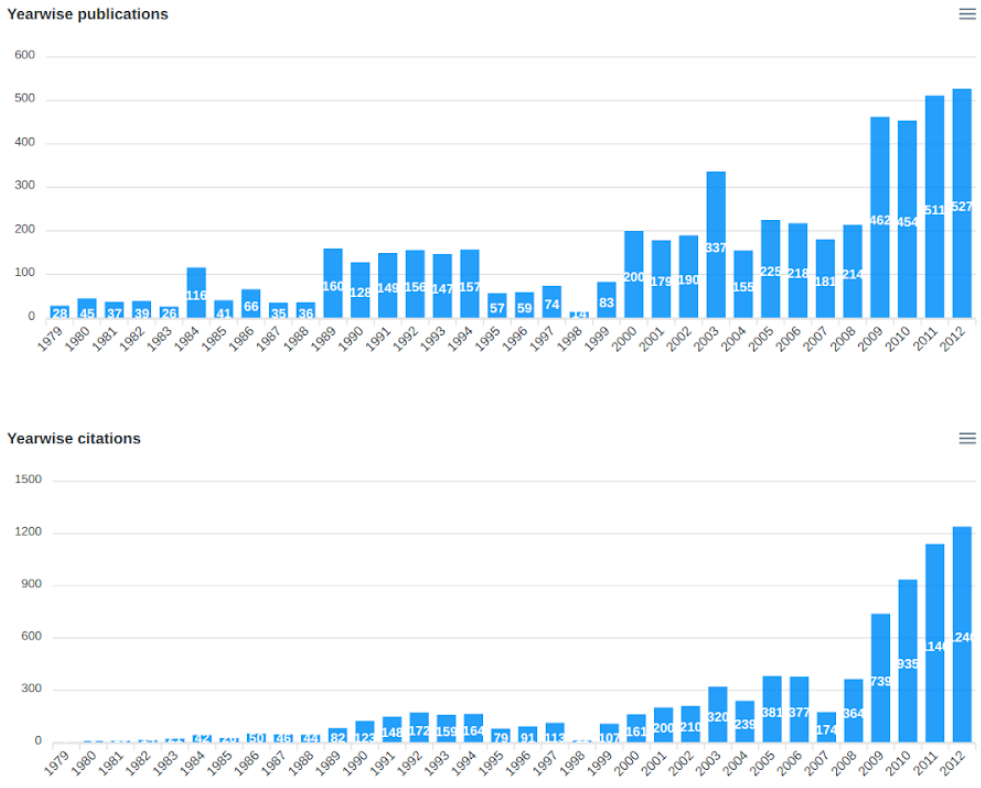
- Coauthor Graphs (max. set to 5)



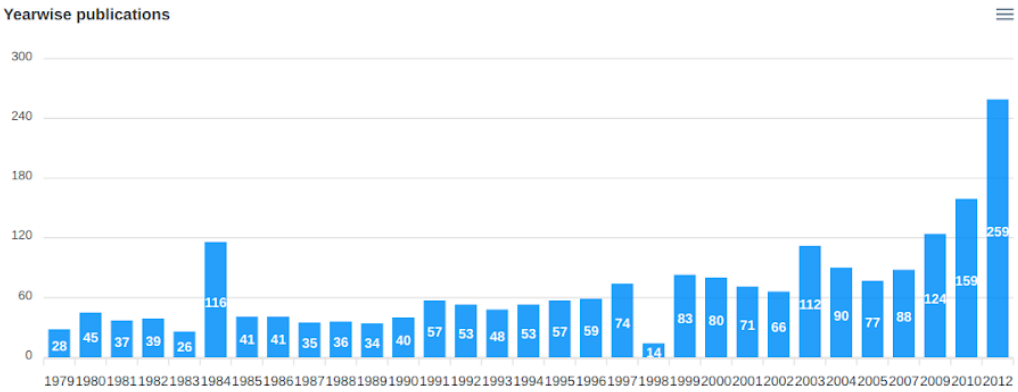
- Yearwise Institute Publications and Citations



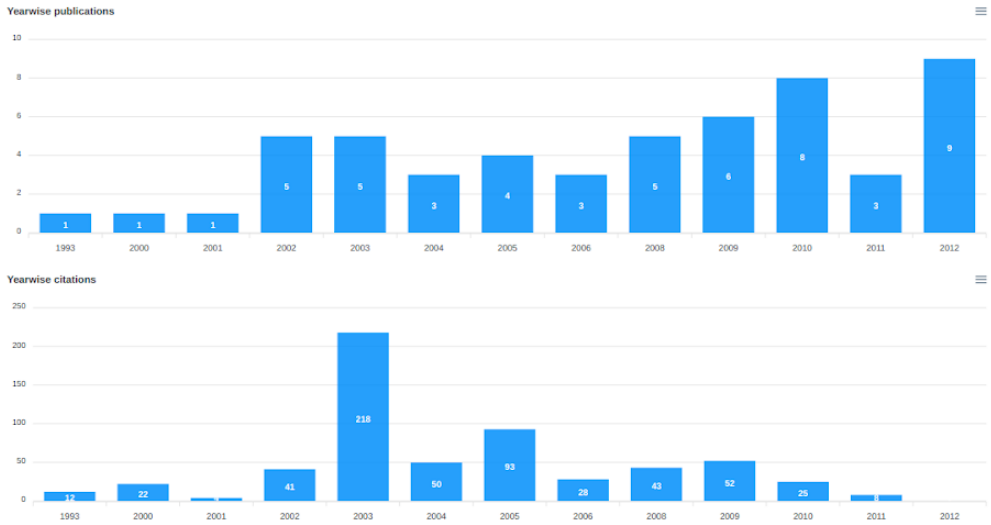
- Yearwise Publications and Citations for a topic



- Yearwise Publications for a Venue



- Yearwise and Topicwise data for authors



Analysis of Work Done/Not Done

One feature which we had initially planned on implementing was regional based queries and performing some analysis on that but since we were unable to get real data with consistent regional data, we had to randomly assign regions to the authors. Thus, the analysis would not be explainable. We therefore decided to ignore it and instead focused on other parts.

We had initially planned to have an interface where the admin can add new articles. However, we missed on the fact that adding new papers means possible addition of new authors, thus potential addition of new institutions.

We had initially planned to be at-par with csrankings.org and csmetrics.org in terms of number of papers and authors. However, upon looking closer at both the open-source websites, we realized that the raw data isn't as good as we had expected and a lot of inconsistencies were present. These inconsistencies were handled by the websites over years by hardcoding several variables and adding filters. We decided on analyzing a subset of the dataset (3 mainstream topics spanning 9 conferences).

One future work we have planned is continuous integration of spark in backend queries. Queries involving filtering seem to be memory intensive due to recomputation of metrics in the backend. This recomputation can be instead done using spark just like it is done for the entire dataset.