Extraction and analysis of subgraphs from online networks

Sai Siva Rohith Tirumalasetti
C00576124
CSCE 590 Project
Spring 2025

Table of Contents

1. Summary

The project aims to develop an application that can extract,visualize, and analyze co-authorship networks from the Dblp bibliographic repository, both from offline XML data and live online sources. Users can search for authors, find collaboration graphs, and derive statistical properties like clustering coefficient and average distance among authors.

2. Objectives

- Provide a fast and scalable system to extract author collaboration graphs from Dblp data.
- Enable users to search for authors and generate visual graphs showing co-author relationships.
- Allow statistical analysis of large-scale collaboration networks.
- Support both offline (static XML dump) and online (live Dblp API) data sources.

3. Background

Dblp provides open access to computer science bibliographic information. With around 3.77 million authors and 7.8 million publications, it is an essential database for bibliometric analysis. The platform will utilize this data for advanced graph analytics and research studies.

4. Scope

Data Ingestion:

- Download and parse Dblp XML files.
- Extract author and publication data.

Backend Application:

- Build RESTful APIs for search, find graph, and statistical analysis.

Frontend UI:

- Web interface for searching and selecting authors.
- Display generated collaboration graphs.

Statistics Computation:

- Degree distribution.
- Clustering coefficient.
- Graph diameter.
- Average path length.

5. Business/Functional Requirements

- Generate a graph representing author collaborations.
- Compute graph statistics from both offline and online data.
- Allow dynamic graph generation for selected authors (online mode).

6. Technical Requirements

- Allow the server to initialize the graph data on startup from the Persons.csv file.
- /find API: Given a list of authors, find common papers and generate collaboration links.
- /search/{name} API: Search Dblp for authors matching a given name.
- UI-based graph visualization after selecting authors.
- Run batch processes to compute statistics such as clustering coefficient and diameter.

7. Tools and Technologies

- Backend: Java, Spring Boot, JAXB
- Frontend: HTML, JavaScript, jQuery
- Build Tools: Gradle
- Other Libraries: https://dblp.org/src/mmdb-2019-04-29.jar
- Deployment Environment: Local Linux server
- Data Source:https://dblp.org/xml/dblp.xml.gz
- XML Schema: https://dblp.org/xml/dblp.dtd

8. Deliverables

8.1 Source code

https://github.com/saisivarohitht/CSCE590

8.2 Deployable Bundle

The **Section 10** demonstrates on how to build the deployable bundle

8.3 Documentation

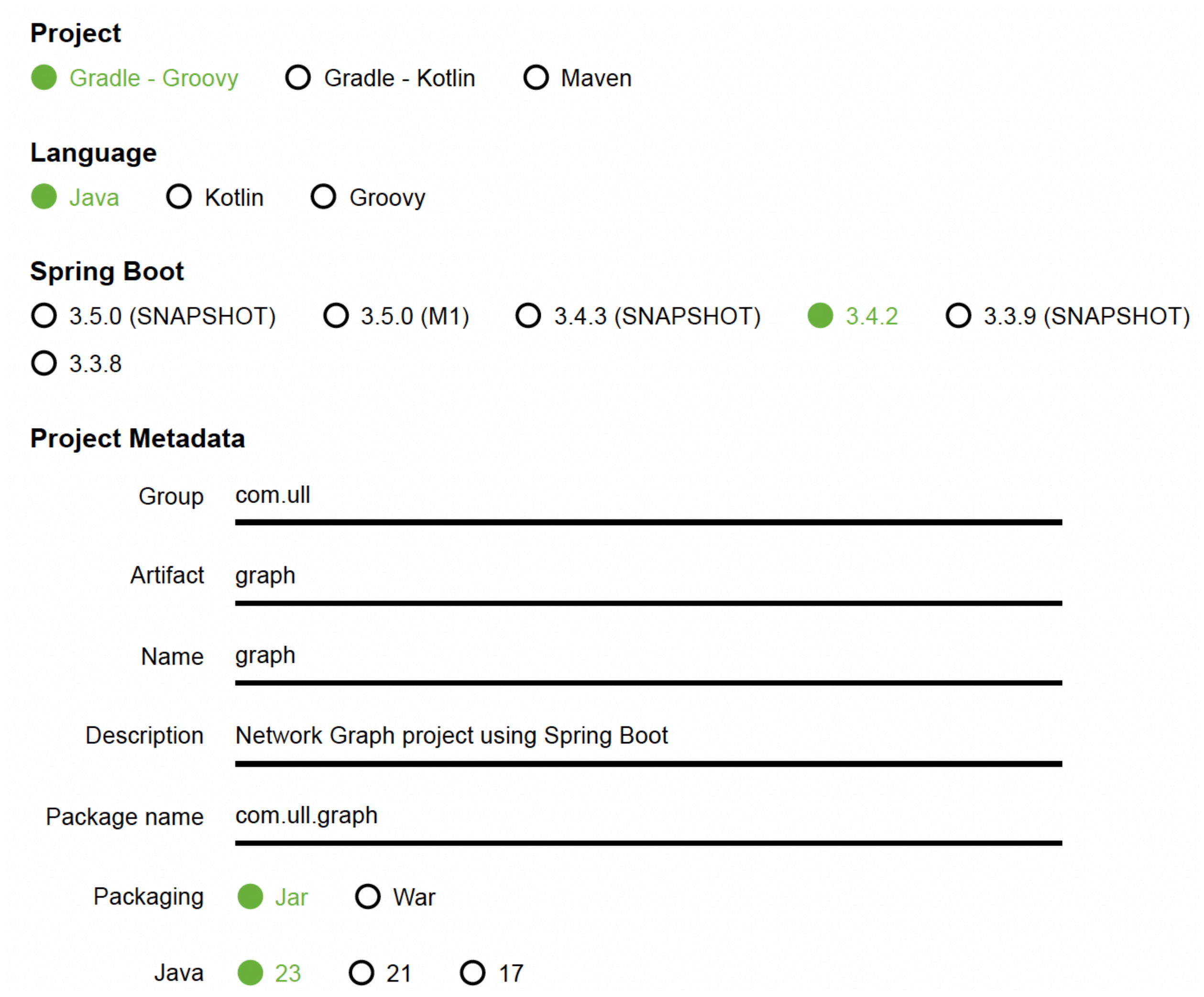
All the documentation related to requirements, APIs and deliverables are covered in this document itself

9. List of Activities completed

- Explore the Dblp repository and gather some knowledge about its contents.
- Identify required software/tools (programming languages, frameworks etc.) and develop a prototype.
- Define and document a set of functional and non-functional requirements.
- Define use cases and the technical design for the system.
- Setting up a development environment.
- Develop a set of RESTful APIs to search / filter data.
- Develop a set of RESTful APIs to generate graphs.
- Develop a Command Line Interface (CLI) to interact with the graph service and display network details.
- Develop a Web UI for interacting with the graph data.
- Integrate of the CLI, UI, and graph service.
- Identify key statistical properties to be evaluated from the Dblp data.
- Develop code to evaluate Statistical Properties.
- Work on completing the documentation.

10. Environment setup

Create a Spring Application from https://start.spring.io/



Project

[✓ Gradle - Groovy] Gradle - Kotlin Maven

Language

[✓ Java] Kotlin Groovy

Spring Boot

3.5.0 (SNAPSHOT) 3.5.0 (M1) 3.4.3 (SNAPSHOT) [✓ 3.4.2] 3.3.9 (SNAPSHOT) 3.3.8

Project Metadata

Group	com.ull
Artifact	graph
Name	graph
Description	Network Graph project using Spring Boot
Package name	com.ull.graph
Packaging	[✓ Jar] War
Java	[✓ 23] 21 17

1. Extract graph.zip file to a folder /home/graph/
2. Update the source code from https://github.com/saisivarohith-tamu/CACS590 to /home/graph/
3. Copy https://dblp.org/src/mmdb-2019-04-29.jar as mmdb.jar to /home/graph/libs/
4. Copy https://dblp.org/xml/dblp.dtd to /home/graph/
5. Copy https://dblp.org/xml/dblp.xml.gz to /home/graph/
6. ./gradlew clean build

11. Generate Authors/Persons data

- Run java -Xmx8g -cp ".:libs/*:build/libs/*" com.ull.graph.controller.GraphController
    - The above command generates the following files
        - /home/graph/Persons.csv (A comma separated file containing Author name and id)
        - /home/graph/AuthorGraph.csv (This would be the input file to generate statistics)
    - Number of publications per each year
    - No of authors
    - No of publications
    - No. of edges in the graph

12. Start Server

Start the Spring Boot server using **java -jar -Xmx8G build/libs/graph-0.0.1-SNAPSHOT.jar**

13. Application Programming Interfaces (APIs)

13.1 /findGraph - find common links/edges among authors provided in input

```
@PostMapping("/find")
public ResponseEntity<Object> findGrah(@RequestBody String[] names) {
```

- The server reads the contents of Persons.csv file and maintains a Map of name, id of all persons (One time activity at the time of start of the server)
- The API reads the id corresponding to each name in the input
- The API builds a URL http://dblp.org/<pid>.xml for each person name in the request and gets the XML response using URLConnection
- The XML response is converted into a java object called Dblpperson using JAXB
- Populate all the publications for each person into a Map
- Find combination of coauthors from the list of all the persons

Request:

```
curl -H "Content-Type: application/json" -X POST -d "[\"Puyan Mojabi\", \"Diana Chirkova\", \"Jim Gray 0001\", \"Ani Thakar\", \"Peter Z. Kunszt\"]" http://localhost:8080/find
```

Response:

```
{
    "nodes": [
        {
            "name": "Puyan Mojabi",
            "id": 1
        },
        {
            "name": "Diana Chirkova",
            "id": 2
        },
        {
            "name": "Jim Gray 0001",
            "id": 3
        },
        {
            "name": "Ani Thakar",
            "id": 4
        },
        {
            "name": "Peter Z. Kunszt",
            "id": 5
        }
    ],
    "links": [
        {
            "source": 1,
            "target": 2
        },
        {
            "source": 3,
            "target": 4
        },
        {
            "source": 3,
            "target": 5
        },
        {
            "source": 4,
            "target": 5
        }
    ]
}
```

13.2 /search/{name} - Search for all the matching person names in Dblp for a given search string.

```
@GetMapping("/search/{name}")
   public List<String> searchMatchingNames(@PathVariable("name") String name) {
```

Request:
```
curl -X GET -H "Content-Type: application/json" http://localhost:8080/search/Jim%20Gray
```

Response:
```
["Jim Gray","Jim Gray 0001","Jim Gray 0002"]
```

Note: When user specify '*' for name, the API returns all authors in the Dblp system

Some of the Dblp statistics are collected by running the following methods from main:

13.3 generateAuthorsGraphFromOnlineDblpDB()

Request: The names of authors are extracted from input.txt

Here is the sample input:

```
Puyan Mojabi
Diana Chirkova
Jim Gray 0001
Ani Thakar
Peter Z. Kunszt
```

Response: The co-authors combination would be extracted into Output.csv file

Here is the sample response:

```
1,2
3,4
3,5
4,5
```

13.4 generateAuthorsGraphFromOfflineDblpDB()

Input: dblp.xml.gz

Output: AuthorGraph.csv (Here are first few lines of 27833895 records):

```
3596080,2954455
3191357,3588946
159787,848508
831366,789217
1746585,1486228
3453302,3557022
```

```
2801750,3682797
172706,3461067
3602992,3350379
```

Note: It takes about 6 hours to generate the graph for the complete offline database

13.5 generatePublicationsCountsPerYearFromOfflineDblpDB()

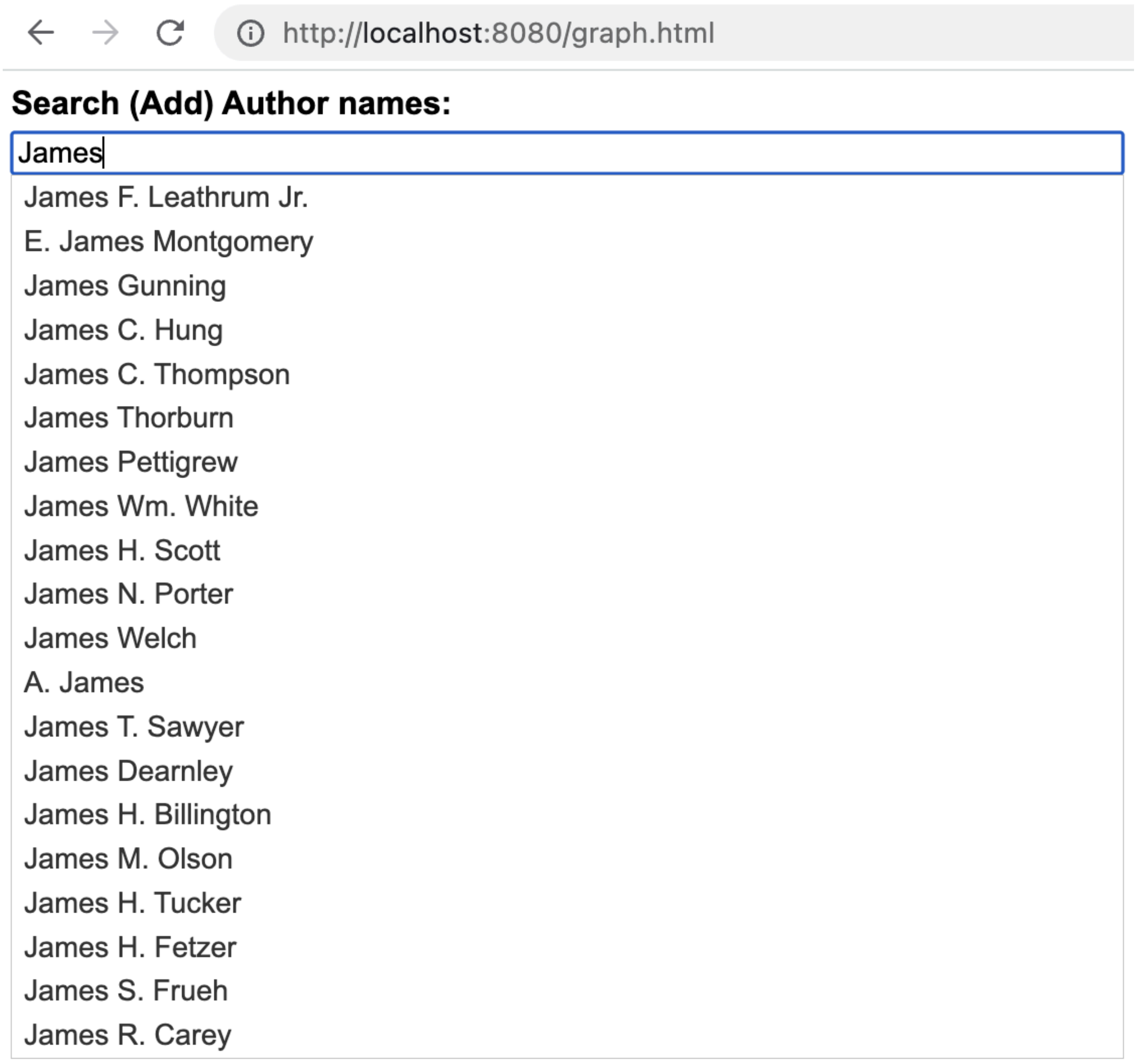
Input: dblp.xml.gz

Output: (Here are the last few lines of output)

```
2015 304427
2016 316727
2017 341660
2018 377849
2019 421137
2020 438054
2021 466329
2022 485313
2023 523375
2024 547923
2025 114466
```

14. UI screens

1. UI can be opened using http://localhost:8080/graph.html
2. UI screen to select the authors for which graph needs to be generated (using AutoComplete feature):



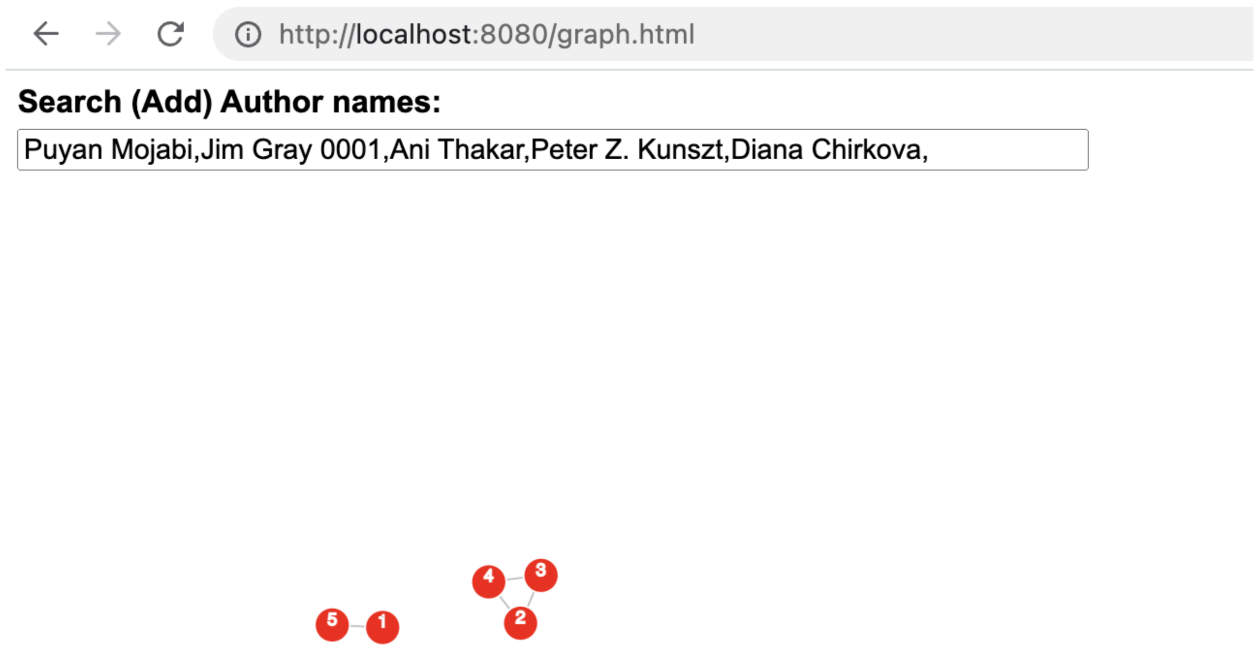
http://localhost:8080/graph.html

Search (Add) Author names:

James

James F. Leathrum Jr.
E. James Montgomery
James Gunning
James C. Hung
James C. Thompson
James Thorburn
James Pettigrew
James Wm. White
James H. Scott
James N. Porter
James Welch
A. James
James T. Sawyer
James Dearnley
James H. Billington
James M. Olson
James H. Tucker
James H. Fetzer
James S. Frueh
James R. Carey

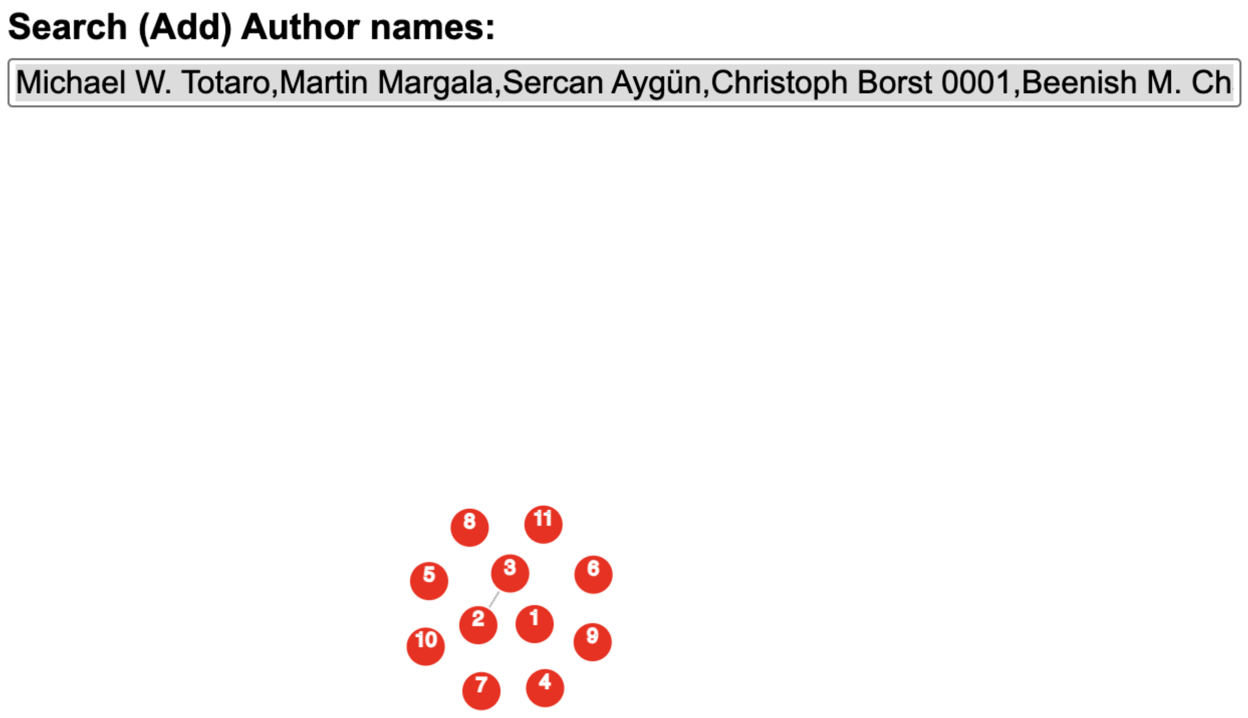
3. Graph: Once all the author names are entered in the above screen, the graph would be generated on pressing the ENTER key


http://localhost:8080/graph.html

Search (Add) Author names:

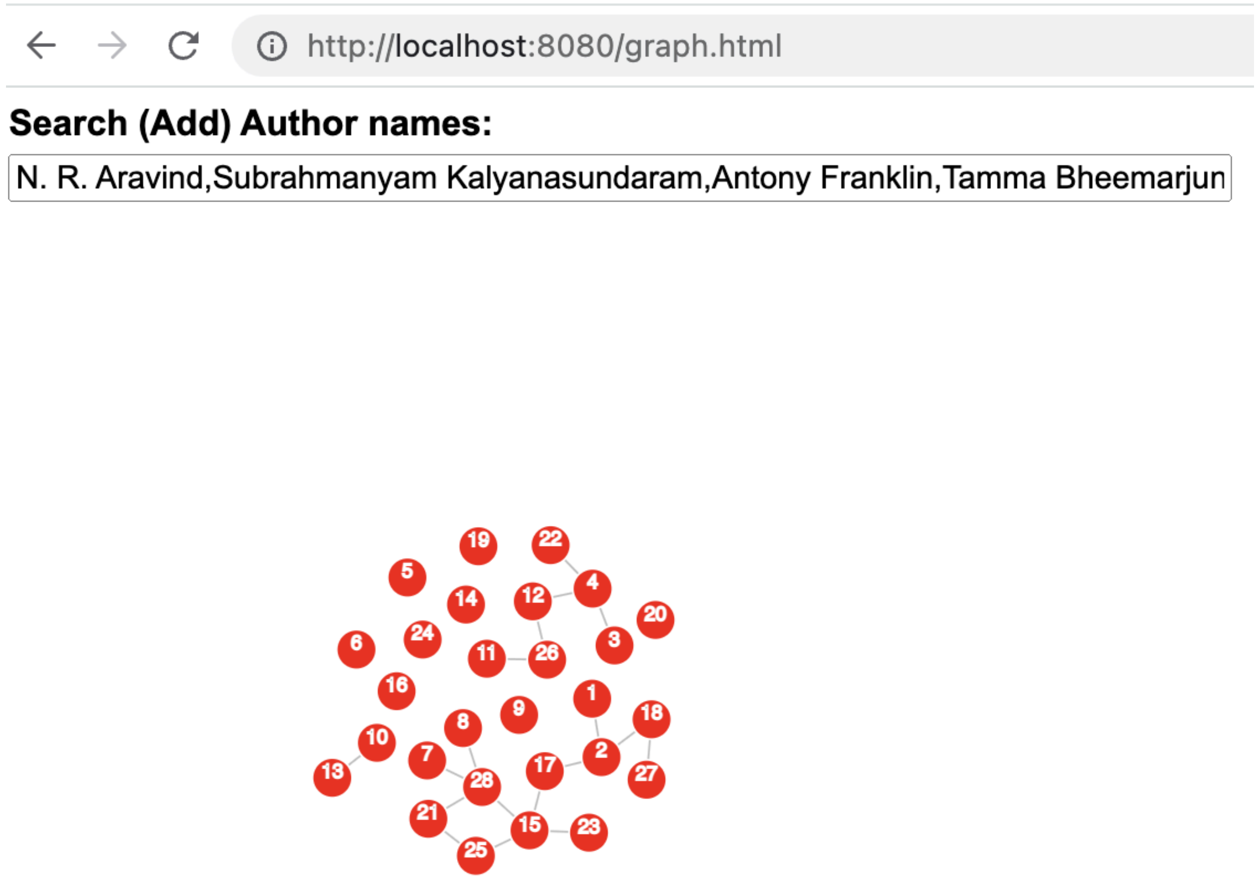
Puyan Mojabi,Jim Gray 0001,Ani Thakar,Peter Z. Kunszt,Diana Chirkova,

Michael W. Totaro,Martin Margala,Sercan Aygün,Christoph Borst 0001,Beenish M. Chaudhry,Chu,Chee-Hung Henry Chu,Shuvalaxmi Dass,Hei,Xiali Hei 0001,Islam,Md. Aminul Islam,Islam,M. Aminul Islam,Arun K. Kulshreshth,


Search (Add) Author names:

Michael W. Totaro,Martin Margala,Sercan Aygün,Christoph Borst 0001,Beenish M. Ch

N. R. Aravind,Subrahmanyam Kalyanasundaram,Antony Franklin,Tamma Bheemarjuna Reddy,Jyothi Vedurada,Saketha Nath Jagaralpudi,Saketha Nath Jagarlapudi,Sakethanath Nath Jagarlapudi,J. Saketha Nath,Karteek Sreenivasaiah,Maria Francis,Praveen Tammana,Nitin Saurabh,Rajesh Kedia,Ramakrishna Upadrasta,Rameshwar Pratap,Sathya Peri,Rogers Mathew,Manish Singh,Sobhan Babu Chintapalli,P. K. Srijith,C. Siva Ram Murthy,Rakesh Venkat,Shirshendu Das,Maunendra Sankar Desarkar,Kotaro Kataoka,Fahad Panolan,Vineeth N. Balasubramanian,



Search (Add) Author names:

N. R. Aravind,Subrahmanyam Kalyanasundaram,Antony Franklin,Tamma Bheemarjun

15. Dblp statistics

The end goal is to derive the following parameters for Dblp data:

- Degree of distribution
  - The degree of a node in a network is the number of connections or edges the node has to other nodes.
- Diameter of giant component
  - In network theory, a giant component is a connected component of a given random graph that contains a significant fraction of the entire graph's vertices. The diameter of such a giant component is to be calculated here.
- Average distance
  - The average distance in a graph is defined as the average length of a shortest path between two vertices, taken over all pairs of vertices.
- Clustering coefficient
  - In graph theory, a clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together.
- No of papers by Year

Dataset used for offline analysis:

https://dblp.org/xml/dblp.xml.gz (2025-04-25 18:15 913M)

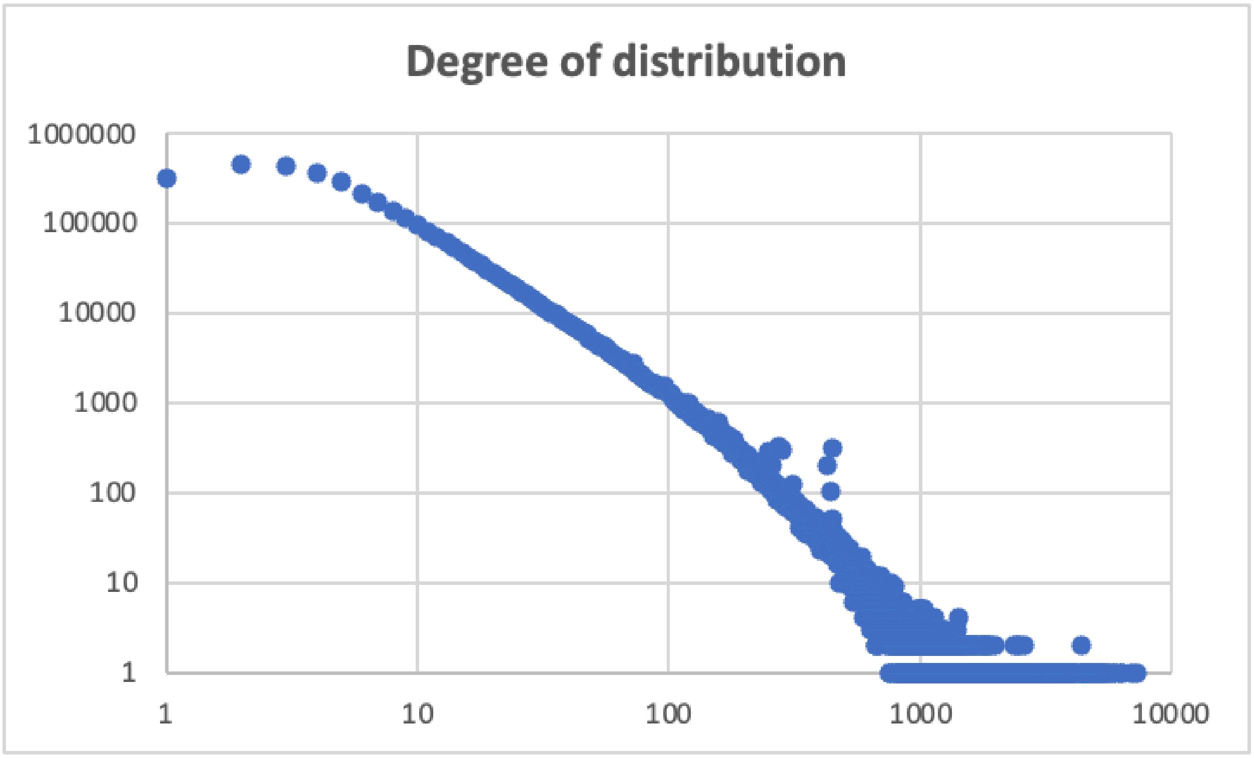
Total number of authors: 3766839   (about 3.77 million)
Total number of publications: 7768302   (about 7.8 million)

Number of vertices	***= 3629419* (over 3.6 million)***
Number of edges	***= 27833895 (about 27.83 million)***
Maximum degree	***= 7263***
Clustering coefficient	***= 0.655***
Diameter of giant component**	***= 19***
Average distance**	***= 4.277***

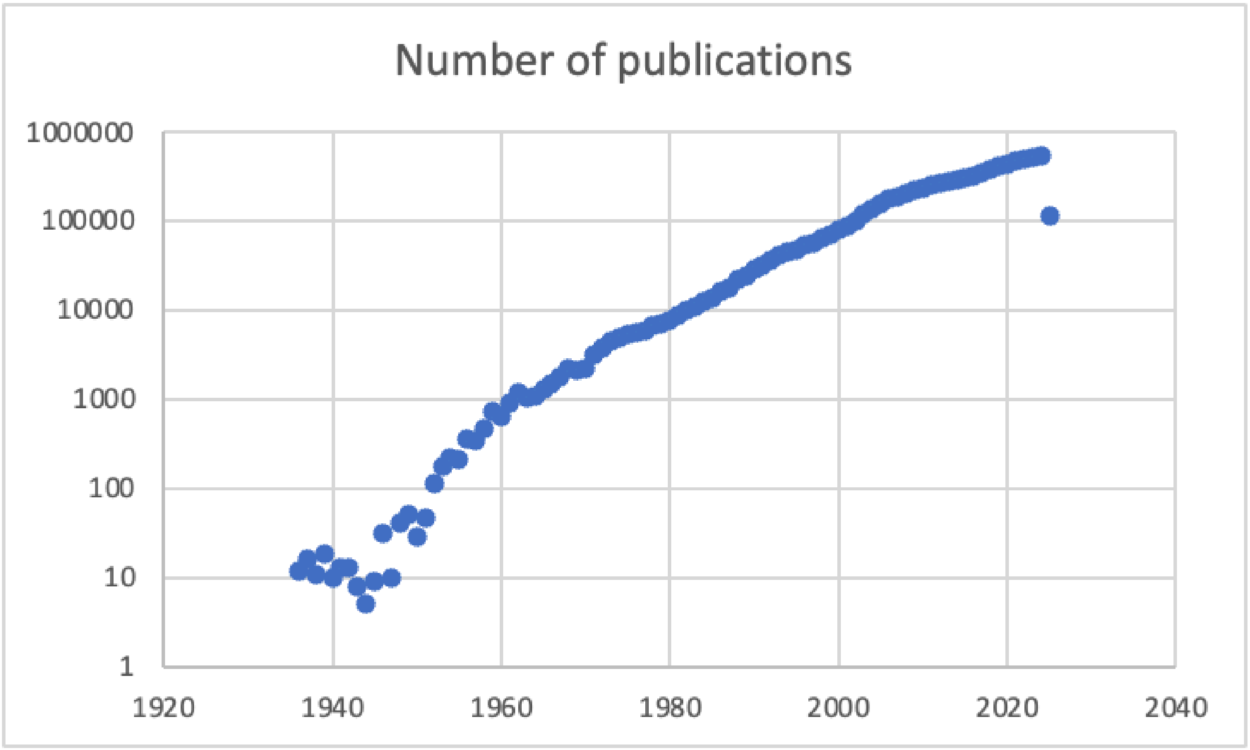
* There are 137420 authors who do not form any edges i.e no collaboration with any other authors.

****** The analysis is limited to about 100000 vertices to calculate these two parameters because of time constraints.

Degree distribution:



No of papers by Year:



16. Conclusions

- In this project, an efficient system was successfully developed for the extraction, visualization, and analysis of co-authorship networks from the Dblp bibliographic repository.
- The system supports both offline data processing (from a static XML dump) and real-time online data retrieval (from live Dblp APIs).
- Key functionalities such as author search, graph generation, and statistical analysis (including degree distribution, clustering coefficient, graph diameter, and average distance) were implemented.
- The backend APIs, along with a user-friendly web interface, enable seamless exploration of author collaboration networks.
- Performance evaluation demonstrated the scalability of the solution, handling millions of authors and publications effectively.
- This work provides a strong foundation for further studies in bibliometric analysis and large-scale graph analytics.

17. Future Work

Several enhancements can be considered to further improve the system:

Scalability Improvements:
Optimize graph generation and analysis algorithms to handle even larger datasets with reduced memory and processing time.

Dynamic Graph Updates:
Incorporate real-time updates from Dblp's live database without needing a full system restart or manual data reload.

Enhanced User Interface:
Improve the frontend with richer graph visualizations (using D3.js, Cytoscape.js) and interactive features like zooming, filtering, and clustering.

Integration with Other Bibliographic Databases:
Extend the system to fetch and merge data from additional sources like Google Scholar, Semantic Scholar, or Scopus.

User Personalization:
Allow users to save searches, graphs, and track authors or topics of interest over time.

Deployment and Cloud Hosting:
Deploy the system on cloud platforms (AWS, Azure, or GCP) to support wider public access and distributed computing capabilities.

18. References

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
https://jqueryui.com/resources/demos/autocomplete/multiple.html
https://gist.github.com/heybignick/3faf257bbbbc7743bb72310d03b86ee8
https://introcs.cs.princeton.edu/java/45graph/