Citation-Driven Paper Retrieval System based on Knowledge Graphs

Wenyi Wu
HDSI
UC San Diego
wew058@ucsd.edu

Kavit Shah
HDSI
UC San Diego
k9shah@ucsd.edu

Jingjing Zhu
HDSI
UC San Diego
jiz296@ucsd.edu

Abstract

This project presents a novel citation-driven paper retrieval system that addresses the challenge of navigating vast academic literature in the field of Data Science. By leveraging a hybrid database architecture combining PostgreSQL for structured metadata and Neo4j for graph-based relationships, we enable efficient discovery of research papers, trend analysis, and exploration of collaboration networks. Our system utilizes the Semantic Scholar API to collect metadata for approximately 130,000 research papers across 20 key Data Science topics, curated with the assistance of GPT-4. We implemented automated ETL pipelines to process JSON API responses into structured CSV files, which are then loaded into our dual database system. The hybrid querying strategy allows for both precise metadata retrieval through SQL queries and contextual relationship discovery via graph traversal algorithms.

1 Introduction

The increasing volume of academic research papers presents a significant challenge for researchers attempting to stay informed, identify crucial contributions, and discover relevant connections within their fields. Navigating this vast landscape to pinpoint key papers, understand their influence through citations, and find potential collaborators can be time-consuming and often inefficient. To address this challenge, this project proposes the construction of a citation-driven paper retrieval system based on a knowledge graph. This system aims to provide a more intuitive and powerful approach to academic research discovery by leveraging the rich information embedded within citation relationships.

The core idea is to build a system that goes beyond simple keyword-based searches, allowing users to explore the intricate network of scholarly work through its citation graph. By understanding which papers cite others, and which papers are cited by a particular work, researchers can gain deeper insights into the evolution of ideas, identify influential publications, and uncover related research they might otherwise miss. This approach promises to enhance the efficiency of literature reviews, facilitate the identification of research gaps, and promote interdisciplinary collaboration.

To realize this vision, the project will employ a hybrid database strategy, utilizing both PostgreSQL for structured data storage and Neo4j as a graph database. PostgreSQL will be used to store the structured metadata of research papers, including details such as titles, authors, publication information, and keywords. This relational database is well-suited for efficient querying and retrieval of paper-specific attributes. Complementing this, Neo4j will store and model the relationships between these entities, particularly focusing on citation links, co-authorship networks, and connections based on research topics and keywords. By representing papers, authors, institutions, and topics as nodes, and their interactions as edges with specific relationship types (e.g., CITES, AUTHORED BY), a rich and navigable knowledge graph will be created.

The primary source of data for constructing this system will be the Semantic Scholar API, which provides comprehensive metadata for academic papers, including titles, authors, and crucial citation and reference information. While the ArXiv API is a valuable source for research paper metadata such as titles, abstracts, authors, publication dates, and categories, it does not provide citation relationships. Therefore, the Semantic Scholar API is essential for building a citation-driven system. In addition to Semantic Scholar, other APIs like OpenAlex and CrossRef can also be leveraged to obtain citation data, offering alternative sources and potentially broader coverage. The project will initially focus on the Data Science domain to ensure the creation of a high-quality and manageable dataset.

The process of building the system involves several key steps, including data collection, processing, storage, and querying. Data collection will utilize the Semantic Scholar API to retrieve raw research paper data in JSON format, ensuring that metadata, citations, and references are captured. This raw data will then undergo processing using Python libraries like ‘json‘ and ‘pandas‘ to extract relevant attributes such as paper titles, authors, citation relationships, and keywords. To ensure consistency and facilitate database integration, each paper will be assigned a unique identifier, and citation links will be explicitly mapped. The processed data will then be transformed into CSV format and loaded into both PostgreSQL and Neo4j. PostgreSQL will store structured data in tables like ‘papers‘, ‘authors‘, and ‘keywords‘, while Neo4j will populate the knowledge graph with nodes and relationships based on the extracted information.

Once the database is constructed, the system will employ a hybrid querying strategy to enable efficient and comprehensive paper retrieval. PostgreSQL will be used for structured queries, such as searching for papers by keywords, authors, or institutions. Subsequently, Neo4j’s graph traversal capabilities will be utilized to explore relationships, such as finding papers that cite a specific work, identifying highly cited papers within a network, or discovering co-authorship connections. This combination of structured and graph-based querying will provide users with a powerful tool for navigating the academic landscape, uncovering hidden connections, and ultimately enhancing the process of research discovery. The user interface will allow for functionalities such as searching for papers, exploring citation networks and co-authorship relationships, and potentially visualizing trends in the Data Science field.

2 Methodology

The methodology for building the citation-driven paper retrieval system involves several key steps. Data collection is the initial phase, primarily utilizing the Semantic Scholar API to gather research paper metadata, including titles, authors, citation relationships, and references. Although the ArXiv API provides paper metadata, it lacks citation information, making Semantic Scholar crucial for this project. The project focuses specifically on the Data Science domain to build a targeted and high-quality dataset.

Following data collection, the raw JSON data is processed using Python libraries like ‘json‘ and ‘pandas‘. This involves parsing the JSON files to extract key attributes such as paper titles, authors, citation links, and keywords. To ensure consistency, each paper is assigned a unique identifier, and citation relationships are mapped to establish connections between papers.

The processed data is then transformed into CSV format for efficient loading into the two chosen databases: PostgreSQL and Neo4j. PostgreSQL stores structured metadata in tables like ‘papers‘, ‘authors‘, ‘keywords‘, and their respective mappings, ensuring efficient querying and retrieval of specific paper attributes. Neo4j, on the other hand, serves as the graph database, modeling relationships such as citations (CITES) and co-authorships (AUTHORED BY). In Neo4j, papers, authors, and keywords are represented as nodes, with the relationships forming the edges of the knowledge graph.

Finally, the system employs a hybrid querying strategy. For structured searches, such as finding papers by keywords, the system utilizes PostgreSQL. For exploring relationships like citation networks and co-author connections, the system leverages Neo4j’s graph traversal capabilities. This combined approach aims to optimize both search efficiency and the discovery of meaningful connections within the academic literature. The implementation utilizes libraries like ‘psycopg2‘ for PostgreSQL and ‘GraphDatabase‘ for Neo4j. Before loading, data cleaning and preparation are performed to ensure consistency.

3 Data Collection

3.1 Overview

Our goal is to gather research paper metadata and citation relationships to construct a structured paper graph for efficient academic paper retrieval. Given the vast number of academic papers across various fields, we focus specifically on the Data Science domain.

To build a high-quality dataset, we extract key attributes for each paper, including:

- **Paper ID**
- **Title**
- **Authors**
- **Keywords**
- **Citation relationships**

To achieve this, we leverage the **Semantic Scholar API**, which provides comprehensive metadata for academic papers. The API returns structured **JSON data**, containing details such as citation count, author names, and references to other papers. This dataset serves as the foundation for constructing our database, enabling efficient modeling of citation relationships.

3.2 Data Collection Process

3.2.1 Topic Selection and Paper Retrieval

To ensure comprehensive coverage, we leveraged GPT-4o to generate a list of 20 key topics in the Data Science field. For each topic, we collected highly cited papers along with their respective citation networks.

As a result, we successfully compiled a dataset of approximately 130,000 research papers, covering a broad range of topics such as Machine Learning, Reinforcement Learning, Graph Neural Networks, and more. The bar chart below visualizes the distribution of collected papers across various keywords, highlighting the most researched areas.

3.2.2 Data Processing

The collected data was initially stored in JSON format and then converted into CSV files for structured storage in databases.

To automate this process, we use **Python**, following a structured three-step pipeline:

Step 1: Data Collection using the Semantic Scholar API We query the Semantic Scholar API to collect raw research paper data in JSON format, ensuring that we retrieve metadata, citations, and references for each paper.

Step 2: Parsing JSON Files Using Python's json and pandas libraries, we iterate through all JSON files and extract key attributes such as: Paper titles, Authors, Citation relationships, Keywords, etc.

To maintain consistency across datasets, each paper is assigned a unique identifier, and citation relationships are mapped to establish connections between papers.

Step 3: Transforming Data into CSV Format Once the data is structured, we convert it into CSV format to facilitate smooth integration with our databases. The CSV files are organized as follows:

- `papers.csv` → Stores paper metadata.
- `authors.csv` → Contains author information.
- `keywords.csv` → Lists extracted research keywords.
- `citations.csv` → Captures citation relationships between papers.
- `paper_authors.csv` → Links papers to their respective authors.
- `paper_keywords.csv` → Maps keywords to relevant papers.

4 Database Construction

4.1 Data Import and Graph Construction

Database Construction For our database setup, we built two complementary systems. In Neo4j, we created a graph structure where papers, authors, and keywords are represented as nodes. Relationships like AUTHORED_BY and CITES link papers to authors and citations, making it easy to trace connections. Meanwhile, in PostgreSQL, we designed a structured relational model with tables for papers, authors, keywords, and their mappings. Foreign keys ensure referential integrity between tables. This combination allows us to efficiently perform both traditional structured queries and graph-based searches depending on the use case.

4.2 Database Integration

To connect PostgreSQL and Neo4j, we used two key technologies:

- **psycopg2** for establishing a connection with PostgreSQL.
- **GraphDatabase** from Neo4j to query relationships between papers.

The system follows a **hybrid querying strategy**, where PostgreSQL retrieves structured metadata, and Neo4j enhances results by finding connections between papers through citations and authorship.

4.2.1 PostgreSQL Connection

PostgreSQL stores structured metadata, including paper titles, authors, and keywords. The connection is established using the following Python snippet:

```
import psycopg2
pg_conn = psycopg2.connect(
dbname="final_project",
user="postgres",
password="your_password",
host="localhost",
port="5432"
)
pg_cursor = pg_conn.cursor()
```

Listing 1: PostgreSQL Connection

This enables querying paper metadata efficiently, such as retrieving a paper ID based on a title.

4.2.2 Neo4j Connection

Neo4j is used to analyze relationships between papers through citations and co-authorship. The connection is established using:

```
from neo4j import GraphDatabase
neo4j_driver = GraphDatabase.driver("bolt://localhost:7687",
auth=("neo4j", "your_password"))
```

Listing 2: Neo4j Connection

This allows us to execute Cypher queries to find papers that cite a given research paper or retrieve co-authors.

5 System Workflow

5.1 Paper Search Workflow

The process of retrieving relevant papers follows these steps:

1. The user searches for a keyword or paper title.

2. PostgreSQL retrieves relevant paper IDs.
3. These IDs are passed to Neo4j to find citation relationships and co-authored papers.
4. The final results include both structured metadata and relationship-based insights.

5.2 Example Queries

Users can utilize this system to explore papers' citation networks and also the same auther's related works. There are some sample workflow: **Retrieving papers by keyword from PostgreSQL:** if given keywords, we first use PostgreSQL to find papers with the keyword.

```
SELECT p.paper_id, p.title
FROM papers p
JOIN paper_keywords pk ON p.paper_id = pk.paper_id
WHERE pk.keyword_id = (SELECT keyword_id FROM keywords WHERE name
    ILIKE '%Machine Learning%');
```

Listing 3: Keyword-Based Paper Retrieval

Digging the citation networks in Neo4j: Given the paper_id, we can explore its citation network in Neo4j. There is an example of finding the top 5 papers in the citation network with the largest citation count.

```
citation_query = """
            MATCH (p:Paper {paper_id: $paper_id})<-[:CITES|CITED_BY]-(
    citing:Paper)
            with citing
            RETURN citing.paper_id AS citing_paper_id, citing.title AS
     citing_title
            ORDER BY citing.citation_count DESC
            LIMIT 5
            """
            result = session.run(citation_query, paper_id=paper_id)
            citations = result.data()
```

Listing 4: Finding Cited Papers in Neo4j

Finding same author's paper in Neo4j: We can also utilize the author-paper relationship and find the related paper:

```
     with neo4j_driver.session() as session:
        query = """
        MATCH (p:Paper {paper_id: $paper_id})<-[:AUTHORED_BY]-(a:
    Author)-[:AUTHORED_BY]->(other_p:Paper)
        WHERE p.paper_id <> other_p.paper_id
        RETURN other_p.paper_id AS paper_id, other_p.title AS title, a
    .name AS author
        ORDER BY other_p.citation_count DESC
        LIMIT 10;
        """
        result = session.run(query, paper_id=paper_id)
        return result.data()
```

Listing 5: Finding Papers Authored by Same author in Neo4j

5.3 Conclusion

The integration of PostgreSQL and Neo4j enables a comprehensive research paper retrieval system that leverages both structured metadata and graph-based relationships. This approach enhances traditional SQL-based searches with citation and co-authorship analysis, making academic research discovery more efficient.

6 Lessons Learned

Based on the project proposals and our conversation, some key lessons learned include the importance of selecting the appropriate data sources. The ArXiv API provides valuable metadata but lacks citation information, highlighting the necessity of integrating with sources like the Semantic Scholar API to capture citation relationships. The project also demonstrated the benefits of a hybrid database approach, leveraging PostgreSQL for efficient structured data storage and querying and Neo4j for effectively modeling and exploring relationships within the knowledge graph. Furthermore, the process of data collection, processing, and transformation into suitable formats for both databases is a critical aspect, requiring careful planning and implementation using tools like Python and specific libraries for each database. Finally, combining the strengths of structured (SQL) and graph-based queries allows for more powerful and nuanced information retrieval.

7 Future Work

Future work could focus on developing a user-friendly web application with a front-end built using React.js or Next.js, potentially incorporating D3.js or Cytoscape.js for visualizing citation networks. Further enhancements could include implementing paper recommendations based on citation patterns, enabling users to identify trending topics in Data Science, and exploring co-authorship networks visually. Integrating tools like NetworkX & PyVis for a more interactive citation graph exploration is another potential direction. These steps would build upon the current hybrid system to make academic research retrieval even more powerful and intuitive.