A Ranking Enabled Scholarly Search System

Junfeng Liu, Shuai Ma, Renjun Hu, Chunming Hu and Jinpeng Huai
SKLSDE Lab, Beihang University, Beijing, China
Beijing Advanced Innovation Center for Big Data and Brain Computing, Beijing, China
{liujunfeng, mashuai, hurenjun, hucm, huaijp}@buaa.edu.cn

***Abstract*—Scholarly search systems greatly aid the deep understanding of scholarly data and facilitate the research activities of scholars for scientific studies. Though a number of such systems have been developed, most of them either support rankings of limited search of entities or provide only basic ranking metrics. These existing systems also mainly adopt RDBMS as their storage such that the linked feature of scholarly data is not fully exploited. In this study, we design and develop a novel scholarly search system Athena. (1) It supports four types of scholarly entity searches: articles, authors, venues and affiliations, and is equipped with five ranking metrics, including three traditional metrics and two comprehensive importance ranking metrics. (2) It also provides profiling of scholarly entities. (3) It further utilizes a graph storage to directly leverage the linked feature for speeding up the processing of complex queries. We shall demonstrate the advantages of Athena from three aspects: scholarly search, author profiling and the graph storage.**

I. Introduction

The continuous advancements in science and engineering have contributed to an ever-growing body of scientific literature. To aid the deep understanding of scholarly data and to facilitate the research activities of scholars for scientific studies, a number of scholarly search systems have been developed, which essentially provide searches and profiling of scholarly entities (articles, authors, venues and affiliations).

Given a query of a specific research topic, Google Scholar[1], Semantic Scholar[2] and CiteSeerX [1] only rank scholarly articles, while AMiner [2] further provides rankings of authors. Also, the supported ranking metrics in these systems are either simple (*e.g.,* time and relevance) or biased to older articles (*e.g.,* citation counts). AceMap [3] and Microsoft Academic[3] do rank heterogeneous scholarly entities, *i.e.,* articles, authors, venues and affiliations. However, AceMap simply ranks those entities based on the numbers of their associated articles with respect to the query, and Microsoft Academic does so by inferring query intent (based on the search log from Bing). To conclude, these existing systems may face with limitations when users perform entity searches.

Besides, storage is another key factor for scholarly search systems, due to the large volume of data and the complex relationships between entities, *e.g.,* Microsoft Academic Graph (MAG) contains 126 million articles and 529 million citations [4]. Observe that scholarly entities are inherently linked, and existing systems mainly exploit RDBMS as their storage solutions. Hence they cannot fully exploit this linked feature, and become inferior when answering complex scholarly queries [5]. For instance, complex joins in RDBMS become a bottleneck when *finding the top-k articles of an author*.

Contributions. In this study, we design and develop a scholarly search system Athena to aid the deep understanding of scholarly data and to facilitate the research activities of scholars for scientific studies.

(1) Athena supports four types of entity searches with five ranking metrics. Besides the traditional ranking metrics (relevance, publish time and citation counts), we incorporate two advanced ranking metrics (importance and relevant importance), which has proven effective on ranking articles [6].

(2) Athena also supports profiling scholarly entities, *e.g.,* author profiling with research interest evolution and affiliation profiling with author and field of study visualization.

(3) Athena utilizes graph database Neo4j[4] for storage. We carefully design a Neo4j schema to model scholarly data as a property graph, and we also incorporate Lucene index to speed up query processing.

(4) We develop a prototype system Athena, and shall demonstrate the advantages of Athena from three respects: scholarly search, author profiling and the graph storage using Neo4j compared with the relational storage using MySQL.

Organization. The rest of this paper is organized as follows. Section II introduces scholarly search and ranking model. Our Athena system is presented in Section III, followed by demonstrations in Section IV and conclusions in Section V.

II. Scholarly Search and Ranking Model

In this section, we first introduce scholarly search, and then present the ranking model that lays its core foundation.

A. Scholarly Search

Our Athena system facilitates the research activities of scholars by providing four types of entity searches as follows.

Article search. Given a set of keywords, Athena returns a sorted list of articles whose titles *contain* the given keywords. In the meantime, it also returns three sorted lists of authors, venues and affiliations associated with the returned articles.

Author, venue and affiliation searches. Given a (partial) name of an author, venue or affiliation, Athena returns a sorted list of authors, venues or affiliations, respectively, that *contain* the given (partial) name.

[1]https://scholar.google.com
[2]https://www.semanticscholar.org
[3]https://academic.microsoft.com
[4]https://neo4j.com

Table I
FUNCTIONS PROVIDED IN Athena AND EXISTING SCHOLARLY SEARCH SYSTEMS

Systems	Scholarly Search				Article-Coupled Ranking				Visual Profiling			
	AR	AU	VE	AF	AR	AU	VE	AF	AR	AU	VE	AF
CiteSeerX	√	√	×	×	√	×	×	×	×	×	×	×
Google Scholar	√	√	×	√	√	×	×	×	×	×	×	×
Semantic Scholar	√	√	×	×	√	×	×	×	×	√	×	×
AMiner	√	√	×	√	√	√	×	×	×	√	×	×
MS Academic	√	√	√	√	√	√	√	√	×	×	×	×
AceMap	√	√	√	√	√	√	√	×	×	√	√	√
Athena	√	√	√	√	√	√	√	√	√	√	√	√

AR, AU, VE and AF stand for article, author, venue and affiliation, respectively.

B. Scholarly Article Ranking

Athena provides five metrics to support article search.

Citation counts and publish time. Scholarly articles are simply sorted based on their citation counts and publish time.

Importance. This metric comes from our recently developed SARank (please refer to [6] for details). The importance of an article is defined as a combination of its prestige and popularity, where its prestige is computed by a novel Time-Weighted PageRank, and its popularity is the sum of citation freshness. To assign those newly published articles reasonable ranking scores, it further assembles the importance of citations, authors and venues to derive the final ranking.

The above three metrics are for *query independent ranking*, and we introduce two metrics for *query dependent ranking*.

Relevance. This metric enables to rank scholarly articles in terms of the semantic correlation between their titles and given queries (keywords). As Word2vec [7] has become the *de facto* standard to capture semantics, we utilize Word2vec to evaluate semantic relevance as follows.

$$rel(a, Q) = \sum_{t \in a.T} \sum_{q \in Q} idf(q) \frac{\mathbf{t} \cdot \mathbf{q}}{||\mathbf{t}|| \ ||\mathbf{q}||}. \quad (1)$$

Here $a.T$ and Q are the sets of words (excluding stop words) in the title of an article a and the query, respectively, $idf(q)$ is the inverse document frequency of a word q, and $\mathbf{t}$ and $\mathbf{q}$ are their corresponding word embeddings.

Relevant importance. The previous importance metric does not consider the closeness of the articles with respect to given queries (keywords). This metric is to rank scholarly articles by combining the semantic relevance and importance metrics. More specifically, we first normalize the computed relevance (resp. importance) score by scaling with the maximum relevance (resp. importance) score in the resulting article set, and we then define the relevant importance score as follows.

$$rImp(a, Q) = \alpha \cdot rel_n(a, Q) + (1 - \alpha) \cdot imp_n(a), \quad (2)$$

where $rel_n(a, Q)$ and $imp_n(a)$ are the *normalized* semantic relevance and importance scores, and α is a regularization parameter, typically set to an empirical value in $[0.2, 0.4]$.

C. Author, Venue and Affiliation Ranking

For authors, venues and affiliations, Athena provides stand-alone ranking and article-coupled ranking.

Stand-alone ranking. To support author, venue and affiliation searches, Athena ranks authors, venues and affiliations with the sum of the query independent ranking scores of all their associated articles. Further, when sorting entities by publish time, it uses an exponentially decayed ranking score $e^{T_a - T_0}$ to evaluate the research activeness of an entity, where T_a is the publish year of an article a, and T_0 is the current year.

Article-coupled ranking. To support article search, Athena ranks authors, venues, and affiliations with respect to the resulting articles of an article search query, along the same lines as stand-alone ranking. This helps to answer questions like which authors (venues or affiliations) are the most authoritative ones in the query related field of study.

The search and ranking functions of Athena and existing systems are summarized in Table I, where metrics are not reported due to their unavailability in commercial systems.

III. Athena SYSTEM

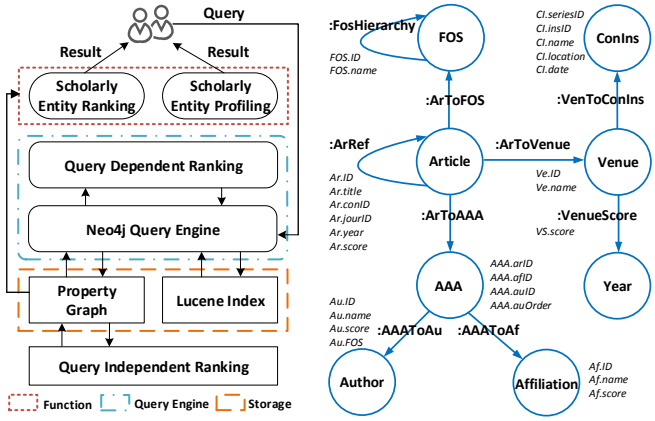
In this section, we introduce our Athena system. As shown in Figure 1(a), Athena consists of three main components, *i.e., storage*, *query engine* and *function modules*. Below the storage component is a *query independent ranking* module that enriches the scholarly data with pre-computed query independent ranking scores, there is another *query dependent ranking* module inside the query engine, and the two function modules support visual analyses for users.

We next explain our system in detail.

A. Schema Design

Graph database Neo4j is adopted for storage in Athena. To do so, we need to design a schema that abstracts the entities and linked structures (*e.g.,* citation, authored-by). We follow two principles for schema design: (1) nodes for entities and relationships for linked structures, and (2) trading space for query efficiency if affordable and possible.

The schema is presented in Figure 1(b), where the texts near nodes and relationships represent the properties of entities and linked structures, respectively. It contains seven basic types of nodes including *Article*, *Author*, *Affiliation*, *Venue*, *FOS* (field of study), *ConIns* (conference instance in each year) and *Year*. In addition, it further incorporates an artificial type of nodes, *i.e., AAA* representing article-author-affiliation tuples. Here we trade extra space for query efficiency, *i.e.,* an author and her/his affiliations can be retrieved in one query. As another space-efficiency trade-off, we also use extra space to maintain certain



(a) Framework (b) Neo4j schema

Figure 1. System design of Athena

properties of *article* nodes: conference ID, journal ID and year. Our schema also forms a total of nine types of relationships, one of which, *i.e., :VenueScore*, has a *score* property.

B. Graph Storage

Following the above schema, we maintain scholarly data as a huge property graph, *e.g.,* the one of MAG [4] has more than 1.03 billion nodes and 1.93 billion relationships.

First, based on the original scholarly data, the *query independent ranking* module pre-computes those query independent ranking scores: citation counts of articles, importance scores of articles, authors, venues and affiliations [6]. These scores are assigned as properties to the corresponding nodes. As iteratively accessing the linked entities is the most essential operation in the pre-computation, the graph storage is much more convenient and effective for such computations. Moreover, both citation counts and importance scores support incremental computation [6], and they are easy to be dynamically maintained once the property graph gets updated.

Second, to facilitate query processing on the billion-scale property graph, Lucene index is utilized for initial entity lookups. Specifically, we create fulltext indices for article titles, author names, venue names and affiliation names. These enable to efficiently find articles, authors, venues and affiliations whose titles or names contain specific keywords.

C. Graph Query Engine

Utilizing Neo4j, Athena supports a variety of graph queries on the property graph to aid scholarly search. And the *query engine* is responsible for processing this set of queries. When a query is issued, the *Neo4j query engine* first translates it into a Neo4j Cypher query with proper parsing and semantic analyses. When applicable, the Cypher query also includes the entity IDs returned from the Lucene index. Relevance and relevant importance rankings given by the *query dependent ranking* module may also be included in the Cypher query if needed. Based on the final Cypher query, an optimized query plan is generated and processed on the property graph.

```
1.  GraphDatabaseService graphDB = new GraphDatabaseFactory()... ;
2.  try ( Transaction tx = graphDB.beginTx() ) {
3.  IndexHits <Node> hits = db.index().forNodes ( "fullTextIndex" )
4.                              .query( "title: data AND title: mining" );
5.  List <ReleImpScore> listScore = calcReleImp(hits, "data mining");
6.  List <String> arIDs = getTopKIDs(listScore);
7.  Result result = graphDB.execute("
8.      WITH {arIDs} AS IDs UNWIND IDs AS perID
9.      MATCH (ar:Article)-[:ArToAAA]->(r)-[:AAAToAu]-(au:Author)
10.     WHERE ar.arID = perID
11.     WITH ar.title, COLLECT(au.auName), SIZE(()-[:ArRef]->(ar)) AS cite, ...
12.     RETURN arID, title, authors, cite, ...");
13.   tx.success();
14. }
```

Figure 2. Example workflow of Athena query engine

Figure 2 gives an example workflow of the query engine when a user wants to search the top scholarly articles about "data mining" ranked by relevant importance. The fulltext index is firstly used to get the related article IDs on "data mining" (lines 3–4). The *query dependent ranking* module then calculates the relevant importance scores of those related articles, and the top-k article IDs are further identified (lines 5–6). Based on the complete Cypher query, the *Neo4j query engine* finally generates the query plan, executes it on the property graph, and returns the results (lines 7–12).

D. Function Modules

Scholarly entity ranking and scholarly entity profiling are the two function modules that collect the ranked scholarly entities returned from the back-end, and present a visual analysis to users. More specifically, Athena utilizes RESTful APIs and Echarts[5] for the scholarly ranking and profiling. Further, Athena provides users with the APIs of the ranking and profiling functions.

IV. Athena Demonstration

We demonstrate our Athena system from three aspects: (1) scholarly search, (2) author profiling, and (3) performance comparison between Neo4j and RDBMS MySQL.

(1) Scholarly search. Figure 3 presents the scholarly entity rankings under query "data mining". The ranking metrics are displayed on the left hand side. Note that Athena also supports queries within a specific range of years. When sorted by relevant importance, the rankings of articles and the associated entities (authors, affiliations, journals and conferences) are shown in the middle and on the right hand side. Further, it is worth pointing out that users can also directly query entities by typing their names in the search box.

(2) Author profiling. Figure 4 gives an example of author profiling for Prof. Jiawei Han. First, the affiliation and number of published articles and citations are presented. Second, a word cloud is given to summarize the author's fields of study, where the size of a word represents its importance in the author's research career. For instance, "data mining" and "machine learning" are identified as Prof. Jiawei Han's most

[5]http://echarts.baidu.com

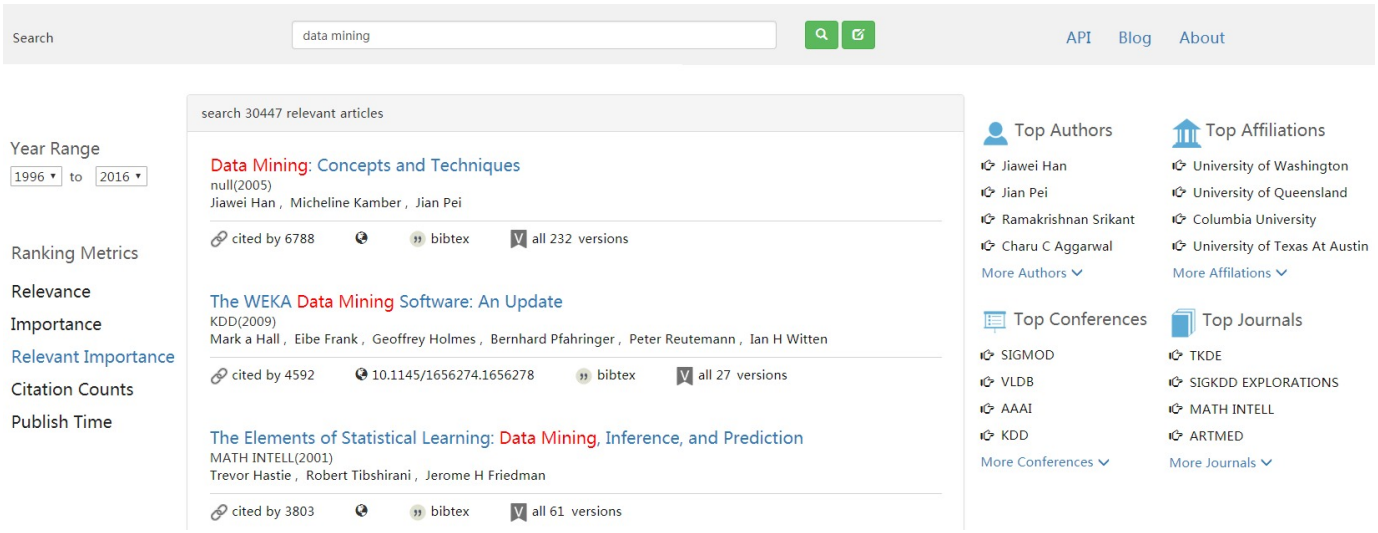


Figure 3. Demonstration of scholarly search

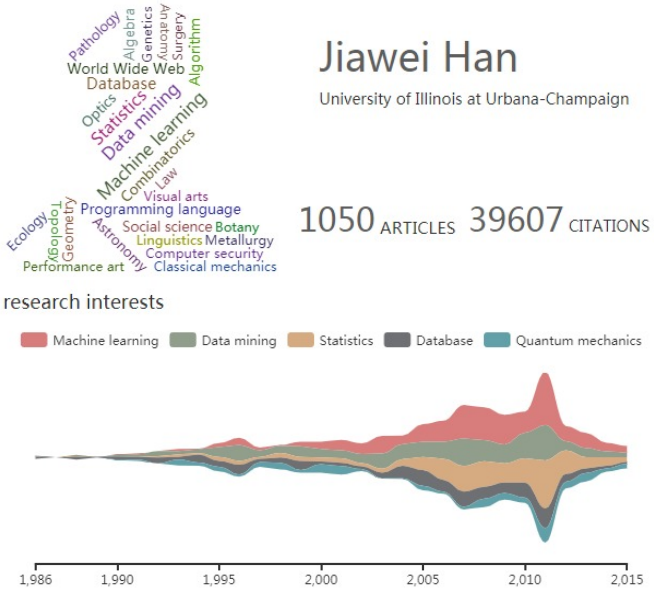


Figure 4. Demonstration of author profiling

important fields of study. Third, Athena illustrates the details of the evolution of research interests, shown at the bottom of Fig. 4, and users can find the relevant articles in each field of study. Finally, functions such as co-authors analyses and statistics by times are provided in author profiling.

(3) Performance evaluation: Neo4j vs. MySQL. Lastly, we compare the query performance of the adopted graph database Neo4j with traditional RDBMS MySQL. More specifically, we consider three types of queries, *i.e.,* (i) given an article ID, finding its title and authors, (ii) given an article ID, finding its top–10 most important citations, and (iii) given an author ID, finding her/his published top–10 most important articles. Note that the numbers of joins are (2, 3, 4) for the three types of queries in MySQL, respectively.

We test a scenario for searching "good" articles. We randomly select 1000 articles and 100 authors whose citation counts and published articles are no less than 100 and 50, respectively, and compute the average processing time. As shown in Table II, Neo4j performs consistently faster than MySQL, which is on average (14.8%, 19.4%, 45.3%) faster for the three types of queries. Thus, structure-aware Neo4j is more efficient than MySQL for such scholarly analyses, especially for complex queries with more join operations.

Table II
DEMONSTRATION OF PERFORMANCE: NEO4J VS. MYSQL

Engines	Type I Queries	Type II Queries	Type III Queries
MySQL	0.115 sec.	2.89 sec.	3.60 sec.
Neo4j	0.098 sec.	2.33 sec.	1.97 sec.

V. CONCLUSIONS

In this paper, we have designed, developed and demonstrated a novel scholarly search system Athena to facilitate the research activities of scholars. Athena has supported rankings of four types of scholarly entities with five metrics, and provided profiling functions to enhance the understanding of scholarly data. Further, Athena has adopted a popular graph database Neo4j as the storage solution for efficient query processing on billion-scale scholarly data.

REFERENCES

[1] H. Li, I. G. Councill, W.-C. Lee, and C. L. Giles, "Citeseerx: an architecture and web service design for an academic document search engine," in *WWW*, 2006.
[2] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: extraction and mining of academic social networks," in *KDD*, 2008.
[3] Z. Tan, C. Liu, Y. Mao, Y. Guo, J. Shen, and X. Wang, "Acemap: A novel approach towards displaying relationship among academic literatures," in *WWW Companion*, 2016.
[4] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-j. P. Hsu, and K. Wang, "An overview of microsoft academic service (mas) and applications," in *WWW*, 2015.
[5] S. Ma, J. Li, C. Hu, X. Lin, and J. Huai, "Big graph search: challenges and techniques," *FCS*, vol. 10, no. 3, pp. 387–398, 2016.
[6] S. Ma, C. Gong, R. Hu, D. Luo, C. Hu, and J. Huai, "Query independent scholarly article ranking," in *ICDE*, 2018.
[7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.