

Graph-based Interactive Bibliographic Information Retrieval Systems

A Thesis

Submitted to the Faculty

of

Drexel University

by

Yongjun Zhu

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

February 2017



© Copyright 2017

Yongjun Zhu. All Rights Reserved.

Acknowledgments

I would like to thank all the people who give me tremendous support and help to make this thesis happen.

First I am deeply grateful to my advisors Dr. Erjia Yan and Dr. Il-Yeol Song, for their generous time and devotion on supervision and guidance in the past years. Dr. Yan has set up a great example for me as a successful information scientist and professor. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. It has been an honor to be his first Ph.D. student. I thank Dr. Song for his guidance and support as the co-advisor and the mentor for my career. His keen insights and vision always helped me explore and study promising research fields and ask important research questions. I always feel lucky to have Dr. Song as my co-advisor. I would like to thank my thesis committee members, Dr. Christopher C. Yang, Dr. Weimao Ke, and Dr. Chaojiang Wu, for their great feedback and suggestions on my thesis. I have also benefited a lot from Dr. Yang and Dr. Ke's courses on Research Statistics and Information Retrieval. Dr. Wu has given me lots of valuable comments during my proposal defense, which improved the thesis.

I would also like to thank my fellow students, including Meen Chul Kim, Hyewon Kim, Qing Ping, Yuanyuan Feng, Bo Song, and Yizhou Zang, who have given me a lot of help and support.

Finally and above all, I owe my deepest gratitude to my parents, my brother, and his wife. I want to thank them for their endless and unreserved love, who have been encouraging and supporting me all the time. This thesis is dedicated to them.

Table of Contents

LIST OF TABLES	ix
LIST OF FIGURES	x
ABSTRACT	xii
1 INTRODUCTION	1
1.1 Motivation and Overview	1
1.2 Research Questions	3
1.3 Contributions	4
1.4 Outline	5
2 RELATED WORK	6
2.1 Graph Models and Information Retrieval Systems	6
2.1.1 Graph Models and Graph Data Stores	7
2.1.2 Graph-based Information Retrieval Systems	9
2.1.3 Bibliographic Information Retrieval Systems	10
2.2 Search Interfaces	12
2.2.1 Natural Language Interfaces	12
2.2.2 Named Entity Recognition	13
2.2.3 Syntactic Analysis	14
2.2.4 Graph Query Interfaces	15
2.3 Evaluation of Interactive Information Retrieval Systems	16
2.3.1 Relationships of Interactive Information Retrieval Measures	16
2.3.2 Work and Search Tasks	17

2.3.3	Interactive Information Retrieval Evaluation Frameworks and Systems ..	19
3	Methodology	21
3.1	Form-based Bibliographic Information Retrieval System	21
3.1.1	System Overview	21
3.1.2	System Design	22
3.1.2.1	System Architecture	23
3.1.2.2	Query Generation	24
3.1.2.3	Query Refinement	25
3.1.2.4	Querying Graph Database	27
3.2	Natural Language-based Bibliographic Information Retrieval System	28
3.2.1	System Overview	28
3.2.2	System Design	28
3.2.2.1	System Architecture	28
3.2.2.2	The Analysis of Natural Language Queries	30
3.2.2.3	The Generation of Graph Queries	32
3.2.2.4	The Translation of Graph Queries	39
3.3	Visual Graph-based Bibliographic Information Retrieval System	41
3.3.1	System Overview	41
3.3.2	System Design	42
3.3.2.1	Bibliographic Graph Queries	42
3.3.2.2	System Architecture	44
3.3.2.3	The Verification of Visual Graph Queries	45
3.3.2.4	The Generation and Interpretation of Graph Queries	49

4	Results.....	53
4.1	Form-based Bibliographic Information Retrieval System.....	53
4.1.1	A System Use Case.....	53
4.1.2	Experiments	55
4.1.2.1	Functional Limitations of Current Bibliographic Information Retrieval Systems	55
4.1.2.2	A Comparison between Graph Data Model and Relational Data Model.....	57
4.2	Natural Language-based Bibliographic Information Retrieval System.....	64
4.2.1	A System Use Case.....	64
4.2.2	Experiments	66
4.3	Visual Graph-based Bibliographic Information Retrieval System.....	69
4.3.1	A System Use Case.....	69
4.3.2	Experiments	73
5	User-centered Evaluation.....	76
5.1	Experimental Setup.....	76
5.1.1	Overall Experimental Setup.....	76
5.1.2	Tasks and Measures	76
5.2	Results.....	80
5.2.1	Participants' Overall Experience	80
5.2.2	Results of Performance Evaluation.....	81
5.2.3	Results of Usability Evaluation	84
6	Conclusion and Future Work.....	89
6.1	Conclusion	89

6.2 Future Work	91
LIST OF REFERENCES	94
APPENDIX A: NATURAL LANGUAGE QUERIES TESTED IN THE EXPERIMENT	103
APPENDIX B: FIFTEEN QUERIES TESTED IN THE EXPERIMENT	106
VITA	108

List of Tables

3.1	Tokenization without NER and with NER	32
3.2	Dependency relations of the query “papers about information retrieval and data mining”	33
3.3	Dependency relations of the query “papers that were written by John”	35
3.4	Graph nodes in the query “papers that were cited by papers that were written by John”	36
4.1	The number of bibliographic entities and relations in the dataset	59
4.2	Meta-paths and example natural language queries	60
4.3	The ratio of correctly answered queries	67
4.4	The ratio of directly representable queries in each system	74
5.1	Two examples of task situation used in the experiment	78
5.2	Performance measures used in the experiment	79
5.3	Results on the performance evaluation	82
5.4	Results of the mixed-design ANOVA	85
5.5	Results of pairwise t-test on system type	86
5.6	Results of ANOVA and Tukey’s tests	87

List of Figures

3.1	A conceptual schema for bibliographic data.....	22
3.2	The process flow of the form-based system	23
3.3	A sample dataset	25
3.4	An information retrieval example.....	26
3.5	The flow chart of the natural language-based system.....	29
3.6	The flow chart of selecting graph relations form dependency relations.....	34
3.7	The check of connectedness and directions of the query “papers by happy university”	37
3.8	The integration of cited and citing parts in the query “authors cited by John”	39
3.9	The translation of the graph query “authors that were cited by John” into a graph query language	40
3.10	Four example visual graph queries	43
3.11	Architecture of the visual graph-based bibliographic information retrieval system	45
3.12	An algorithm for query semantics checking	46
3.13	Examples of query correction and disambiguation.....	47
3.14	An example of generating candidate graph queries.....	50
3.15	Examples of translating visual graph queries into Cypher	51
4.1	Generate Cypher query	54
4.2	The search results for the example query	56
4.3	An ER diagram for the relational database-based system	58
4.4	Query execution time of queries of different groups.....	61

4.5	Average query execution time of queries	63
4.6	A natural language interface with an example query.....	64
4.7	The analysis of a natural language query.....	65
4.8	The search results of the example query.....	66
4.9	The query execution time of queries with the number of named entities from two to five	68
4.10	The visual graph interface of the system	70
4.11	Candidate graph queries and search results	71
4.12	A semantically incorrect visual graph query	72
4.13	Two candidate graph queries and search of the query	73
5.1	Descriptive statistics for usability measures	85

Abstract

Graph-based Interactive Bibliographic Information Retrieval Systems

Yongjun Zhu

Erjia Yan, Advisor, Ph.D.

Il-Yeol Song, Co-advisor, Ph.D.

In the big data era, we have witnessed the explosion of scholarly literature. This explosion has imposed challenges to the retrieval of bibliographic information. Retrieval of intended bibliographic information has become challenging due to the overwhelming search results returned by bibliographic information retrieval systems for given input queries. At the same time, users' bibliographic information needs have become more specific such that only information that best matches their needs is seen as relevant.

Current bibliographic information retrieval systems such as Web of Science, Scopus, and Google Scholar have become an unalienable component in searching bibliographic data. However, these systems have limited support of complex bibliographic queries. For example, a query- "*papers on information retrieval, which were cited by John's papers that had been presented in SIGIR*" is an ordinary information need that researchers may have, but is not appropriately representable in these systems. In addition, these systems only support search for papers and do not support other bibliographic entities such as authors and terms as the final search results.

Therefore, in this dissertation, we propose several bibliographic information retrieval systems that can address complex bibliographic queries. We propose form-, natural language-, and visual graph-based systems that allow users to formulate bibliographic queries in a variety of ways. The form-based system allows users to formulate queries by

selecting forms and input values in those selected forms. In the natural language-based system, users formulate queries using a natural language. Users formulate queries by drawing nodes and links in the visual graph-based system. These systems are based on a graph model to enhance retrieval efficiency and provides interfaces for users to formulate queries interactively.

Through a system-centered evaluation, we find that our graph-based system took less time to process complex queries than a relational-entity-based system (two secs vs. several mins on average). In addition, our visual graph-based system can deal with the representation of advanced queries such as bibliographic coupling, paper co-citation, and author co-citation, while current bibliographic information systems do not support these queries. A user-centered evaluation reveals that participants rated the natural language-based system the most useful, easy to use, and easy to learn. Participants also reported that the form-based system was easier to learn than the visual graph-based system. Based on the results of a usability evaluation, we find that the form-based system is preferred for low-complexity tasks while the visual graph-based system is preferred for high-complexity tasks. The strength of the natural language-based system is that no additional effort is needed to formulate more complex queries. The proposed systems are effective and efficient solutions for addressing complex bibliographic information needs. In addition, we believe the experimental design and results shown in this paper can serve as a useful guideline and benchmark for future studies.

1. Introduction

1.1 Motivation and Overview

Journal articles and conference papers have proliferated in recent years. This is partly due to online accessibility to scholarly literature. Online accessibility has shortened publication cycles by enabling scholars to access others' works before formal publication. Thus, researchers are able to expedite their research activities and publish papers more frequently. A natural concern is that it has become more challenging to find relevant papers and discover knowledge from these papers.

Bibliographic information retrieval systems such as Web of Science, Scopus, and Google Scholar have become an unalienable component in searching bibliographic data (Chadegani et al., 2013). These systems continuously index ever-increasing scientific literature, thus providing a source for scholars to learn, create, and represent new knowledge (Jacso, 2005). However, with such a large amount of scientific literature, sifting through them in hopes of excavating that one applicable nugget of information we yearn for can be daunting and often frustrating—so much time consumed by mining so many thousands of articles. It is a question of ascertaining relevancy. For example, a query, “*papers on information retrieval, which were cited by John’s papers that had been presented in SIGIR*” is an ordinary information need that researchers may have. Given three pieces of background information: (a) SIGIR is a top venue for information retrieval research; (b) John, a well-known researcher, has presented a few papers in SIGIR; and (c) John’s papers were influenced by a number of other papers, researchers,

who found John's papers helpful, might also be interested in his cited papers. With the dramatic increase of scientific literature, there is the pressing need of building effective and efficient bibliographic information retrieval systems that support more granular and complex bibliographic information needs.

A recent work by de Ribaupierre (2014) discussed a few important challenges regarding bibliographic information retrieval in the big scholarly data area: the difficulty of answering scholars' precise bibliographic questions and the lack of techniques that help scholars directly target their information needs. A follow-up interview showed that researchers are interested in searching for papers as well as other entities such as authors and keywords (de Ribaupierre, 2014). By exploring the current bibliographic information retrieval systems, we found these systems have not mastered this. They have two main limitations: (a) a limited support of entity types as the final search results; and (b) a lack of support of complex queries as a way of representing information needs. Accordingly, these systems are not adequate in addressing users' more specific information needs and impose a burden on them to use time consuming post hoc refinements and filtrations.

Motivated by addressing these limitations, we propose graph-based interactive bibliographic information retrieval systems to provide more efficient and effective ways of searching bibliographic information. The proposed systems are efficient because they are based on the graph data model for fast retrieval. The systems are effective because they provide novel ways of formulating bibliographic queries and satisfy specific information needs that are not addressable in the current bibliographic information retrieval systems. Users can develop their queries interactively by referencing the system-generated graph queries.

1.2 Research Questions

To address aforementioned limitations of current bibliographic information retrieval systems, we aim to answer the following research questions.

1) *How to design bibliographic information retrieval systems that support bibliographic queries with complex relations of bibliographic entities?*

To enable effective and efficient traversal through complex relations of bibliographic entities, the choice of underlying data model is critical. The design of the overall system framework that connects users' bibliographic information needs with bibliographic data serves an important research question.

2) *How to design search interfaces that enable the representation of specific and complex bibliographic queries?*

Traditional search interfaces of bibliographic information retrieval systems are limited in their capacity of representing complex bibliographic queries. How to enable users to represent complex bibliographic queries through search interfaces and interpret them is an integral part of the study.

3) *How to implement and evaluate the systems using both system and user-centered approaches?*

Implementation of the back-ends of the systems, query analysis components, and search interfaces is a way to validate the proposed systems. In addition, user-centered evaluations of the systems provide empirical evidence on the applicability of the systems in the real-world environment.

1.3 Contributions

The proposed graph-based bibliographic information retrieval systems are scalable, interactive, and time-efficient for retrieving bibliographic information. They have novel features that outperform other systems: the designed systems support searching for various types of bibliographic information such as papers, authors, affiliations, terms (e.g., keywords), and sources (e.g., journals or conferences); they provide interactive interfaces for users to formulate complex and granular context-based queries; they allow users to modify queries by showing graphical representations generated from users' original searches; and they expedite the all-important retrieval time by adopting the graph data model.

The proposed systems can be used together with other bibliographic information retrieval systems by utilizing each system's advantages. Current bibliographic information retrieval systems such as the Web of Science have advantages in providing rich statistics such as citation, impact factor, and journal information. The rich information can guide users to perform a more informed retrieval in our systems.

The systems can be used to aid researchers and practitioners by finding scholarly literature more easily and quickly. Besides scientific literature, users can also identify prominent researchers and top venues by forming naturally appropriate queries. They also boost users' efficiency in gathering information on researchers and research organizations because they are capable of answering complex queries in a single step. This targeted question-answering aspect satisfies a variety of users' information needs in their expedited quest for relevancy. The systems provide different user interfaces to cater different user preferences.

1.4 Outline

The remainder of this dissertation is organized as follows. Chapter 2 discusses previous work related to our study. Chapter 3 introduces methodologies of the study and present three bibliographic information retrieval systems. In Chapter 4, we present system use cases and the results of a system-centered evaluation. In Chapter 5, we evaluate three systems and report results of a user-centered evaluation. Finally, we conclude this dissertation and introduce future research in Chapter 6.

2. Related Work

This section introduces previous work related to various components of our study. We first introduce fundamental and general knowledge on graph-based bibliographic information retrieval systems such as graph models, graph data stores, graph-based information retrieval systems, bibliographic information retrieval systems. We then introduce several building blocks of the proposed systems such as natural language interface, named entity recognition, syntactic analysis, and graph query interface. Finally, we introduce related work on evaluation of interactive information retrieval systems.

2.1 Graph Models and Information Retrieval Systems

Graph data are prevalent in the real world as data from a variety of domains (e.g., physics, chemistry, biology, sociology, and computer science) can be represented by graph data models (Aggarwal & Wang, 2010). Graph data models can represent relational information and enable a number of applications by supporting efficient searching and mining (Cook & Holder, 2006). Because of this, a few studies have investigated ways of generating graphs from arbitrary data (e.g., Baeza-Yates, Brisaboa, & Larriba-Pey, 2010). Bibliographic data are graph data in nature because they can be represented in the form of interconnected papers, authors, terms, sources, and organizations. Recent bibliometric studies, including searching bibliographic data, measuring scholarly impact (Yan & Ding, 2009), and mining bibliographic networks (Sun, Barber, Gupta, Aggarwal, & Han, 2011) have taken the advantage of the graphical representation of bibliographic data. Regardless

of the physical representations (e.g., relational databases) of graph data, efficient searching of graph data is one of primary tasks for the information retrieval community (e.g., Kacholia et al., 2005; Jiang, Wang, Yu, & Zhou, 2007; Yuan, Wang, Chen, & Wang, 2013).

2.1.1 Graph Models and Graph Data Stores

Graph models have been widely used to represent data types that comprise entities and relations among entities. Graph models have been adopted by various online social networking services such as Facebook and Twitter to represent people and their relations (Sakr & Pardede, 2012). A graph model is suitable to represent domains where many complex relations exist and relations are extremely important to understand the domains (Cook & Holder, 2006). Such domains include the World Wide Web, social networks, biochemical networks, bibliographic information, and power grids. Graph models are ideal to represent bibliographic networks of entities such as papers, authors, terms, sources, and affiliations as well as relations such as cites (i.e., between papers), writes (i.e., between authors and papers), has (i.e., between papers and terms), publishes (i.e., between sources and papers), and affiliated with (i.e., between authors and affiliations).

There are three main ways to represent a graph model, including relational databases, triple stores, and graph databases. Relational databases, such as Oracle, allow users to manage graph data by providing network data models. This type of relational database stores connectivity information in a node table and a link table. Although relational databases support a way of representing graph data, Aggarwal and Wang (2010) pointed out that relational databases are fundamentally inadequate for supporting graph data. In

relational databases, some operations such as graph traversal are costly to implement and the situation becomes even worse as the graphs get larger. A triple store is another popular data store for graph data. A triple store maintains triples and a triple comprises subject, predicate, and object. A triple is generally represented by the Resource Description Framework (RDF; Miller, 1998), which is a standard of W3C. Thus, a triple store is often referred as a RDF store. Among triple stores, Apache Jena (Carroll et al., 2004) and Sesame (Broekstra, Kampman, & Van Harmelen, 2002) are the popular ones. Because triple stores are developed for graph data, they are more powerful than relational databases. However, it is known that triple stores have scalability issues and the performance of triple stores is negatively affected as the number of triples grows (Aggarwal & Wang, 2010). Readers can refer to Rohloff and colleague's study (2007) on the comparisons of different triple store technologies.

Graph databases are the most recent development of graph data stores. It is a category of the NoSQL system that is scalable and supports advanced features such as replication and fault tolerance. Compared with triple stores, graph databases have these advantages (Angles & Gutierrez, 2008): (a) graph databases support the representation of undirected and weighted graphs whereas triple stores only support directed and unweighted graphs; (b) graph databases do not require schema and have the so-called schema-free or schema-less character whereas triple stores explicitly require schema; (c) graph databases are suitable for managing big data. Because of these advantages, graph databases are more suitable for real-world systems that deal with a large amount of data.

2.1.2 Graph-based Information Retrieval Systems

Traditional information retrieval systems have adopted relational databases as the primary way of managing data (Manning, Raghavan, & Schütze, 2008). In 2001, Berners-Lee and colleagues proposed the concept of semantic web for effectively utilizing web resources by adding meanings to web pages. Later, Guha and colleagues (2003, p. 702) coined semantic search by defining it as “*an application of the Semantic Web to search which attempts to augment and improve traditional search results by using data from the Semantic Web.*” Because semantic web technologies are the core component of semantic search, semantic search systems use triple stores as the underlying database. Semantic search systems identify semantic entities from a keyword or natural language-based query, match semantic entities with ontology resources, and then express the meaning of the original query by supplementing it with additional semantic information from an ontology physically represented in a triple store. Thus, queries can be semantically interpreted to deliver more accurate and meaningful results. Among many semantic search systems, SemSearch (Lei, Uren, & Motta, 2006), OntoLook (Li, Wang, & Huang, 2007), SPARK (Zhou, Wang, Xiong, Wang, & Yu, 2007) as well as the system proposed by Tran and colleagues (2007) are the most representative. They differ in terms of query type (i.e., keyword-based vs. natural language-based), multiple semantic matching (i.e., between semantic entities and ontology resources), connections among semantic entities (i.e., direct connection vs. indirect connection), and multiple properties (i.e., one property vs. multiple properties among semantic entities).

Although graph databases are another popular graph data store, to our best knowledge, there lacks a fully designed and developed graph database-based information retrieval

system. However, there are studies that explored issues of graph databases to improve the quality of information retrieval. These issues include, notably, graph database indexing (e.g., Williams, Huan, & Wang, 2007), pattern match query (e.g., Zou, Chen, & Özsu, 2009), subgraph mining (e.g., Huan, Wang, Prins, & Yang, 2004), and substructure similarity search (e.g., Yan, Yu, & Han, 2005). Recently, Internet companies such as Google, Twitter, and Facebook are adopting graph databases for efficient information retrieval (Rajbhandari, Shah, & Agarwal, 2012). For example, Google has adopted “Knowledge Graph”—a form of graph database—to provide better search results (Singhal, 2012).

2.1.3 Bibliographic Information Retrieval Systems

Bibliographic information includes information such as papers, authors, terms (e.g., keywords), sources (e.g., journals, conferences), and affiliations. Google Scholar and the Web of Science only support search for articles as the final search results. They may return articles with metadata such as authors in the result page, but not authors as the final search results (i.e., a list of authors). In order to retrieve all authors’ names, we have to do download and extract metadata from articles in search results. This is labor-intensive given the amount of articles in search results. In addition, other bibliographic information such as “organization” or “keyword” is not directly available in Google Scholar or the Web of Science. We need a bibliographic information retrieval system that supports the search of all pertinent bibliographic information. If users become interested in new, previously unnoticed, but necessary bibliographic information, the system should also be scalable by supporting the search of other bibliographic information. For example,

in some cases, users may want to get terms as the final search results to know which terms are actively studied given a list of articles. Bibliographic information, such as authors and organizations, are important in research evaluation and impact assessment. This bibliographic information can be used to evaluate scientific productivity and impact of authors or their organizations to appropriately make promotion decisions or allocate research funds (e.g., Geuna & Martin, 2003).

Another important feature that bibliographic information retrieval systems need to provide is searching bibliographic information by contexts. With contexts, we mean a variety of ways in which we use related metadata to describe the target bibliographic information. For example, in the case that we search for authors who are affiliated with Happy University and wrote papers that were cited by papers in SIGIR, the system should be able to provide a way in which we can express this context.

Through personal experiences as well as studies about the current bibliographic information retrieval systems (e.g., Aghaei Chadegani et al., 2013; Falagas, Pitsouni, Malietzis & Pappas, 2008; Jacso, 2005; Score, 2009), we found that these systems, such as the Web of Science, Scopus, and Google Scholar only support articles as the final search results. Additionally, these systems have limitations in representing complex bibliographic queries. For example, searching for experts regarding specific terms within an organization is not supported. This kind of query is closely related to many use cases. A student whose research interest is information retrieval, and wants to apply to “University A” may need to identify professors whose interest is also information retrieved and affiliated with “University A.”

2.2 Search Interfaces

2.2.1 Natural Language Interfaces

Natural language interfaces (NLI) are used to query structured information stored in databases. Two types of NLI can be distinguished: one is natural language interfaces to databases (NLIDB), in which a relational database is used to store structured information; the other is natural language interfaces to knowledge bases (NLIKb) that use an ontology to manage information (e.g., Habernal & Konopik, 2013; Abacha & Zweigenbaum, 2015). While the two types of NLI use different database systems, they have common components, including the interpretation of natural language queries and concept mappings between entities in queries and databases (e.g., Cafarella & Etzioni, 2005; Tablan et al., 2008).

The relational data model (Codd, 1970) proposed in the early 1970s had a major impact on NLIDB research. NLIDB are highly portable and can be attached to existing databases because relational databases are the norm of most traditional information retrieval systems (Vicknair et al., 2010). Compared to NLIDB, NLIKb have a relatively short history with the inception of semantic web (Berners-Lee et al., 2001). Databases in this category deploy rich expressive power of ontologies represented in the resource description framework (Miller, 1998), thus generally achieving higher performances (e.g., Kaufmann & Bernstein, 2010). Readers can refer to Androutsopoulos and colleagues' work (1995) for a comprehensive review of NLIDB systems. Recent NLIKb systems include PowerAqua (Fazzinga & Lukasiewicz, 2010), ORAKEL (Cimiano et al., 2008),

FREyA (Damljanovic et al., 2010), PANTO (Wang et al., 2007), and NLP-Reduce (Kaufmann et al., 2007).

Another type is NLI to graph databases (e.g., Roy & Zeng, 2013). Graph databases have comparable expressive power with ontologies (i.e., triple stores), but a much higher scalability, which are more suitable to real-world systems (Angles & Gutierrez, 2008). Graph databases have been increasingly used in information retrieval systems (e.g., Park & Lim, 2015). Graph databases excel relational databases in answerable questions due to its advantage on representing complex relations among data given that natural language queries are represented using complex relations among concepts.

2.2.2 Named Entity Recognition

Named entity recognition (NER) is a task of identifying names of things in texts. These things include but not limited to persons, organizations, locations, and biomedical entities (Nadeau & Sekine, 2007). Early NER systems used rule-based methods to recognize named entities. In a rule-based NER system, patterns in a text are identified and appropriate rules are handcrafted based on those patterns. Thus, a rule-based method is mainly used in self-contained domains and has a limited applicability (e.g., Rau, 1991). A dictionary-based NER system utilizes predefined dictionaries and performs a look-up in texts (e.g., Ryu, Jang, & Kim, 2014; Mu, Lu, & Ryu, 2014). The method is widely used in domains such as biomedicine, in which named entities are well recorded and managed, for instance, in protein recognition (Tsuruoka & Tsujii, 2003) and drug recognition (Rindfleisch et al., 2000). Another popular category of NER is statistical NER (e.g., Derczynski et al., 2015). Widely used statistical NER includes maximum entropy

(ME)- (Chieu & Ng, 2002), hidden Markov models (HMM)- (Bikel et al., 1997), and conditional random fields (CRF)-based (McCallum & Li, 2003) NER systems. Some NER systems use more than one type of NER: for example, Stanford NER (Finkel et al., 2005) provides both dictionary- and statistical-based NER through a gazette feature.

Bibliographic data are relatively easy to obtain through well-known bibliographic databases such as Web of Science and DBLP. Thus, in this dissertation, we used a dictionary-based approach to recognize bibliographic named entities (i.e., authors, papers, organizations, terms, and sources) from a natural language query. By recognizing bibliographic named entities in a query, we are able to extract these entities as well as their relations to learn and answer queries.

2.2.3 Syntactic Analysis

A classic way of parsing is to derive parses from a string of words based on a structure grammar of prewritten phrases (i.e., context-free grammar) (e.g., Earley, 1980). With the introduction of annotated data such as The Penn Treebank (Marcus et al., 1993), a number of statistical parsers were proposed and became popular. Readers can refer to Collins' work (1997) for a more extensive review on statistical parsing models.

Two popular ways of representing syntactic structures are constituency and dependency. For constituency, words in a sentence are organized into nested constituents; while for dependency, dependent relations between words are shown (Klein & Manning, 2004). Dependency parses can be obtained from dependency parsers (e.g., Fersini et al., 2014) or phrase structure parsers (i.e., constituency) by a conversion system (e.g., De Marneffe et al., 2006). In this dissertation, we use a dependency structure to identify

grammatical relations among words. Because we are interested in grammatical relations among bibliographic named entities recognized in natural language queries, dependency structures are more straightforward than constituency structures that also show relations between phrases.

2.2.4 Graph Query Interfaces

Earlier studies on visual graph queries were carried out by taking a specific data structure—XML in mind (e.g., Ceri et al., 1999; Erwig, 2003; Ni & Ling, 2003; Ykhlef & Alqahtani, 2009). These studies proposed visual graph queries for querying and restructuring XML data. As XML data are quite complex with multiple nested structures, visual graph queries are seen as an efficient solution. Because the main goal of these studies was to build efficient languages of visual graph queries by investigating the structural aspects of XML documents, they are intended to be used by other systems but not the end users.

Recent studies (Hogenboom, Milea, Frasincar, & Kaymak, 2010; Schweiger, Trajanoski, & Pabinger, 2014) proposed visual graph query interfaces for users to query graph data. However, these visual graph queries were designed only to search for data that are stored as Resource Description Framework (RDF) triples, which is a standard data format of Semantic Web. Because SPARQL is the de facto standard RDF query language, those visual graph queries were designed to be translated into SPARQL, which limits their applicability. Gómez-Villamor and colleagues (2008) proposed a bibliographic exploration tool based on a graph query engine. The tool employed visual

graphs, while the actual queries are formulated by clicking one of three predefined queries other than a graph.

2.3 Evaluation of Interactive Information Retrieval Systems

Evaluations of interactive information retrieval (IIR) systems have been discussed in studies from earlier decades (e.g., Salton, 1970) to more recent years (e.g., Borlund, 2016). Kelly's two seminal studies (Kelly, 2009; Kelly & Sugimoto, 2013) reviewed extensive studies on this topic published before 2010. Readers may refer to the abovementioned studies to get a detailed understanding of the field. In this section, we provide a review of related work that was published after 2010 to deliver recent findings. In the reviewed literature, we identified three main research themes: studies that explored relationships among established IIR measures; studies that explored a variety of aspects of work and search tasks; and studies that proposed IIR evaluation frameworks and systems. In the following paragraphs, we synthesize the findings of these studies.

2.3.1 Relationships of Interactive Information Retrieval Measures

Al-Maskari and Sanderson (2010) examined the relationship between four factors (i.e., system effectiveness, user effectiveness, user effort, and user characteristics) and user satisfaction to understand whether user satisfaction is influenced by these factors. The authors found a strong correlation between user effectiveness and user satisfaction. System effectiveness and user effort had weak correlations with user satisfaction. However, their results showed no correlation between user characteristics and user

satisfaction. In their subsequent study (Al-Maskari & Sanderson, 2011), the effect of user characteristics on user effectiveness was investigated. User characteristics was measured by users' search experience and cognitive skills (i.e., perceptual speed). A few empirical findings were reported: experienced users retrieved much more relevant documents than inexperienced users and users received higher scores on the perceptual speed test took much less time than users with lower scores to locate the first relevant document. Smucker and Jethani (2010) examined the relationship between retrieval precision and perceived difficulty. They showed that a higher retrieval precision reduced users' perceived difficulty and their relationship was statistically significant.

2.3.2 Work and Search Tasks

Li and Belkin (2010) explored the relationship between work tasks and users' search behaviors in interactive information retrieval. Six types of work tasks were employed based on their faceted classification scheme (Li & Belkin, 2008). They found that users presented different patterns of search behavior in different types of work tasks. Key findings include: in schoolwork-related tasks, users consulted library resources much more often than search engines such as Google, and in decision-making work tasks, users relied more on browsing than in schoolwork-related work tasks. Liu and colleagues (2010) also explored search behaviors in different task types in journalism. They classified tasks based on a modified version of Li's classification scheme (2009) by adding a new facet. Specifically, they examined the associations between search behaviors (e.g., task completion time, number of pages visited, number of queries) and different facet values (i.e., task product, task complexity, level, and task goal). A list of significant associations

were reported in their study. Wildemuth and Freund (2012) reviewed 51 studies on exploratory search tasks and provided a list of task characteristics in exploratory search. Identified task characteristics were grouped into two categories: cognitive (e.g., learning and investigation as goals, general rather than specific) and behavioral (e.g., open-ended, target is multiple items). In a follow-up study, Wildemuth and colleagues (2014) reviewed 106 studies to examine how search task complexity and difficulty were defined and practiced. They identified three dimensions of task complexity including multiplicity of subtasks, multiplicity of facets, and uncertainty. They showed that search task difficulty can be measured both objectively and subjectively. Kelly and colleagues (2015) examined search tasks using a cognitive complexity framework from education theory. Search tasks were created and divided into five levels based on the framework to understand the differences among tasks. They found that participants showed more search activities in more cognitively complex tasks, but did not see more cognitively complex tasks as more difficult. This finding showed that self-reported task difficulty was not in line with physical effort (e.g., queries formulated, clicks, and time taken to complete tasks). Borlund and Schneider (2010) reviewed 85 individual studies that applied the concept of simulated work task introduced by her and Ingwersen (1997) to find how and where it was used. The main goal was to understand how the three parts of recommendations made in her earlier study of IIR evaluation model (Borlund, 2003) were practiced. Results showed that those recommendations were not well applied and studies varied in the way of reflecting and practicing the recommendations. A detailed follow-up study was conducted in 2016 (Borlund, 2016). In addition to the previous findings, the author showed that none of reviewed studies used pilot testing that is

important to tune and refine simulated work task situations. Along with these findings, recommendations were made regarding the design and the creation of simulated work tasks. Clemmensen and Borlund (2016) studied order effect in IIR evaluation. They examined nine IIR parameters of search behavior in a between-group design. Results showed that order effect was evident in three parameters including website change, visit of webpage, and formulation of queries.

2.3.3 Interactive Information Retrieval Evaluation Frameworks and Systems

A few frameworks and systems for IIR evaluations were proposed to aid researchers to conduct systematic evaluations. Renaud and Azzopardi (2012) proposed a web-based system that allows simple within-subjects experiments. The system includes a series of experimental components such as participant registration, consent, surveys, and logging information. While their system was particularly designed for undergraduate and Master's students as the authors stated, Hall and Toms (2013) proposed a framework that provides a common baseline by including existing evaluation measures to enable cross-study comparison. Another goal was to make the framework flexible and applicable to a large number of experiments. To achieve this, the framework defined a standardized set of questions and was designed to be easily integrated with existing systems. Zuccon and colleagues (2013) proposed a method of evaluating IIR systems using a crowdsourcing platform. The proposal was based on the motivation to address a few limitations of traditional laboratory-based evaluations such as high cost and the lack of heterogeneity among the user population. In the proposed method, evaluations of IIR systems including task assignment and data collection are taken online. A case study was performed to

compare the laboratory- and crowdsourcing-based evaluations. Results showed that the two methods led to similar conclusions on the effectiveness of the studied IIR systems while the crowdsourcing-based method could collect five times more data than laboratory-based method with only half the cost. However, a few limitations of crowdsourcing-based method were also pointed out. These limitations include participants' lack of interactions with systems (e.g., issue fewer queries, click fewer documents), the control of data quality (e.g., data generated by bots), and the assessment of the reliability of participants' personal information. Recently, Wei and colleagues (2014) proposed a new web-based system for IIR evaluation. The system enables usability testing of different search task interfaces as well as different algorithms with the same search task interface. One strength of the system is the rich support of eye-tracking logging that includes automated recoding of interface element coordinates. Because the system was primarily designed using JavaScript, it has a broad applicability in the web environment.

3. Methodology

This section discusses methods of the study. We introduce system architectures and components of the systems. Specifically, in the form-based system, we introduce methods on query generation, query refinement, and querying graph database. In the natural language-based system, methods on the analysis of natural language queries, the generation of graph queries, and the translation of graph queries are discussed. Finally, in the visual graph-based system, we discuss methods on the verification of visual graph queries and the generation and interpretation of graph queries.

3.1 Form-based Bibliographic Information Retrieval System

3.1.1 System Overview

The form-based interface is widely used in current bibliographic information retrieval systems including Web of Science and Scopus. In the form-based bibliographic information retrieval systems, users first select fields (e.g., topic and author) and then type appropriate values for each field. In the proposed form-based system, users formulate queries by interacting with forms in a way similar to that of Web of Science. The difference is that the system allows for the formulation of more complex queries that involves citations. A user can select the bibliographic type he search and add additional information to both the cited and citing sides. A graph query is generated based on the form query and allows the user to validate the initial query. Users can modify form queries interactively before sending them to the database.

3.1.2 System Design

The form-based system is developed on the Spring Framework that uses a graph database Neo4j for the management of bibliographic data and D3.js for visualization. Bibliographic data are by nature a directed graph with nodes and links. For example, a link named “WRITES” is a directed link, in which the source is an “Author” and the target is a “Paper”. There are a variety of ways to model bibliographic data using graphs. The one shown in Figure 3.1 shows a typical schema of bibliographic data with five bibliographic entities. In the schema, “Source” denotes a journal or a conference in which authors publish or present papers. “Term” denotes a keyword, a topic; or a concept that describes a paper.

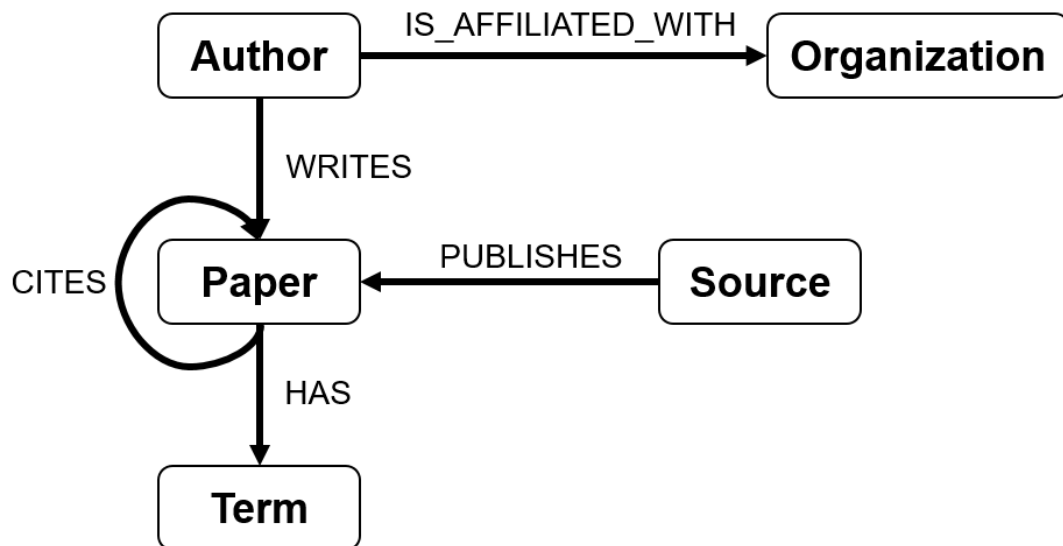


Figure 3.1: A conceptual schema for bibliographic data

3.1.2.1 System Architecture

The system has three parts: query generation, query refinement, and querying graph database. The general steps are shown in Figure 3.2: (a) users generate form queries based on their information needs; (b) the system represents generated form queries in the form of graph queries; (c) generated graph queries are sent back to users; (d) based on the returned graph queries, users refine their queries if necessary; (e) after refinement (optional), form queries are translated into graph database query languages; (f) the system queries graph database; (g) graph database sends query results to the system; (h) the system returns search results to users. The searching process is an interactive and iterative process in which users proceed towards the right representation of their information needs.

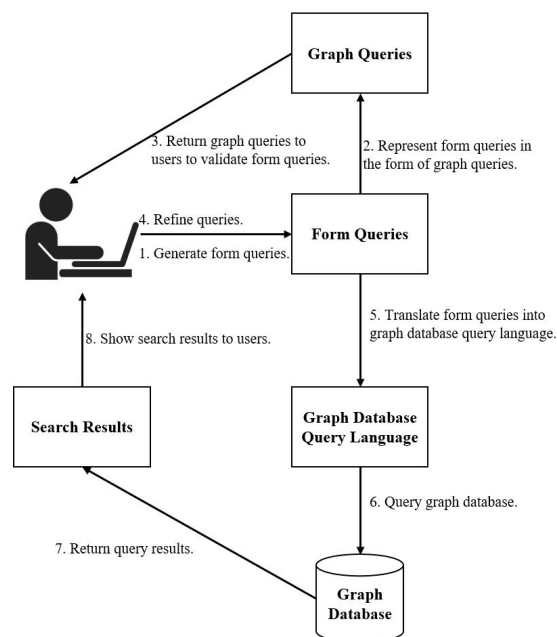


Figure 3.2: The process flow of the form-based system

3.1.2.2 Query Generation

As shown in Figure 3.1, the system supports the search of five types of bibliographic information: papers, authors, terms, sources, and organizations. The system can also support other types of bibliographic information by simply adding and connecting them with the existing entities in the schema. Each type has a property called name. First, the user selects one target entity. Next, the user selects one or a few anchor types that restrict the target entity by providing the values of name properties for each anchor type. For example, when searching for papers, the user can choose a few anchor types such as author and source to restrict the papers.

In scenarios that involve citations, the user needs to designate whether the target is at the cited side or at the citing side. For example, if the user wants to search for authors in the context of citations, the user needs to specify whether the authors are the authors of a cited paper or a citing paper. Then, the user selects anchor types for the cited or citing side. For example, a user who wants to search for authors who are affiliated with Happy University and wrote papers that were cited by papers in SIGIR, needs to select authors as the target type, specify its citation type as cited, add an anchor type—organization to the cited side, and add another anchor type—source to the citing side.

After generating a form query by selecting target types and anchors types, and providing values for anchor types, the form query is sent to the system to generate a graph query.

3.1.2.3 Query Refinement

The system generates graph queries based on users' form queries. The purpose of generating graph queries is to provide users an easy way of verifying their original queries and make modifications if necessary. A graph query is a representation of a form query using nodes and relations. It is generated directly from form queries and shows graph representations of form queries. Figure 3.3 uses a sample data set to showcase the functions of the system.

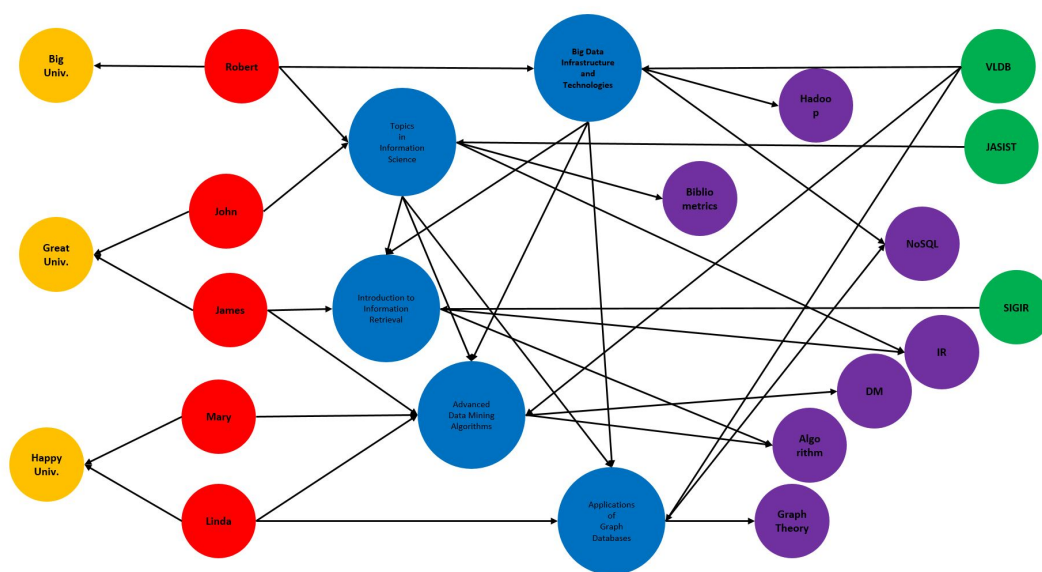


Figure 3.3: A sample dataset

Figure 3.4 shows the interface of a graph query page using the sample data set. As shown in Figure 3.4, relations among various types of bibliographic entities are explicitly shown. The target type is colored black, one or more anchor types with user-provided

labels are colored red. Entities that are not specified by users, but necessary to connect the target and anchors are in blue.

The screenshot displays the GIBIR web application interface, which is divided into three main sections: Form Query, Graph Query, and Search Results.

Form Query: This section contains a form for creating a query. It includes a "Citation" dropdown with options "N/A", "Cited", and "Citing". The "Target" is set to "Author". Under "Anchor", there are two "Add a cited anchor" buttons. The first anchor is "Affiliation" with the name "Happy University". The second anchor is "Term" with the name "NoSQL". There is also a "Source" dropdown with the name "VLDB". A "Generate Graph Query" button and a "SEARCH" button are located at the bottom of the form.

Graph Query: This section shows a graph visualization of the query. The graph consists of several nodes and edges. The nodes are: "Citing Paper" (blue), "Term (NoSQL)" (red), "Source (VLDB)" (red), "Author" (blue), "Affiliation" (blue), "Cited Paper" (blue), "Source" (blue), and "Author (Happy University)" (red). The edges represent relationships between these entities.

Search Results: This section displays the results of the search. It shows "Total: 2 results." and lists "Linde" and "Mary".

Figure 3.4: An information retrieval example

Based on graph queries, users can capture and verify the meaning of the generated form queries. This process is interactive and users can refine their original queries iteratively if necessary. The graph query component was developed by using D3.js. Using forms to represent complex searching context may not be very intuitive; providing interactive interface by showing graph queries can eliminate any confusion that may arise during the formulation of form queries. The step of generating graph queries is optional,

which means users can let the system directly generate graph query language based on form queries.

3.1.2.4 Querying Graph Database

Once the user confirms form queries, the system translates form queries into the graph query language. The system uses Neo4j as the database to represent networks of bibliographic information and Cypher as the query language. Dominguez-Sal et al. (2010) have compared Neo4j with other graph databases, and concluded that Neo4j is one of the two most efficient graph databases with DEX (now known as Sparksee). Cypher is the default query language of Neo4j, and a recent study performed by Holzschuher and Peinl (2013) reported that Cypher has high readability, maintainability, as well as efficiency in development time. Even though the system uses Cypher as a graph database query language, other graph query languages such as SPARQL and Gremlin can be used as long as the graph database in use supports these query languages.

The system translates form queries into Cypher by interconnecting provided bibliographic entities based on the conceptual schema shown in Figure 3.1. When translating form queries into graph query language, connections among bibliographic entities that are not directly shown in the form query are explored and supplemented to form a complete path of the graph query language. After translation, the generated Cypher query is directly sent to Neo4j. Retrieved query results are then returned to the user.

3.2 Natural Language-based Bibliographic Information Retrieval System

3.2.1 System Overview

A natural language interface allows users to formulate queries expressed in natural language. The natural language-based system interprets bibliographic queries expressed in controlled natural language and returns relevant bibliographic data and relations. Natural language queries supported in the system are restricted to complex nominal phrases that describe bibliographic entities. A natural language-based system tailored for bibliographic environment provides a new and effective way of searching bibliographic data. In addition, from practical aspects, by enabling users to formulate bibliographic information needs in natural language, it liberates users from learning cumbersome ways of representing those needs. With ever-increasing bibliographic data, a natural language-based system allows an effective retrieval of data by enabling the representation of complex bibliographic information needs and simplifying the search process into a single step without multiple refining procedures.

3.2.2 System Design

3.2.2.1 System Architecture

The system architecture is designed to take a natural language query as the input and return correct answers as the output. This is achieved by translating the input into a database query language. A natural language query is translated into a graph query language because we use a graph database to manage bibliographic data. Multiple steps

are involved in the translation, including finding answers to questions such as: 1) what is being asked? 2) what entities should be used to constrain the answer? and 3) how does the asked entity relate to other entities? Figure 3.5 uses a flow chart to describe how the core components of the framework interact with each other.

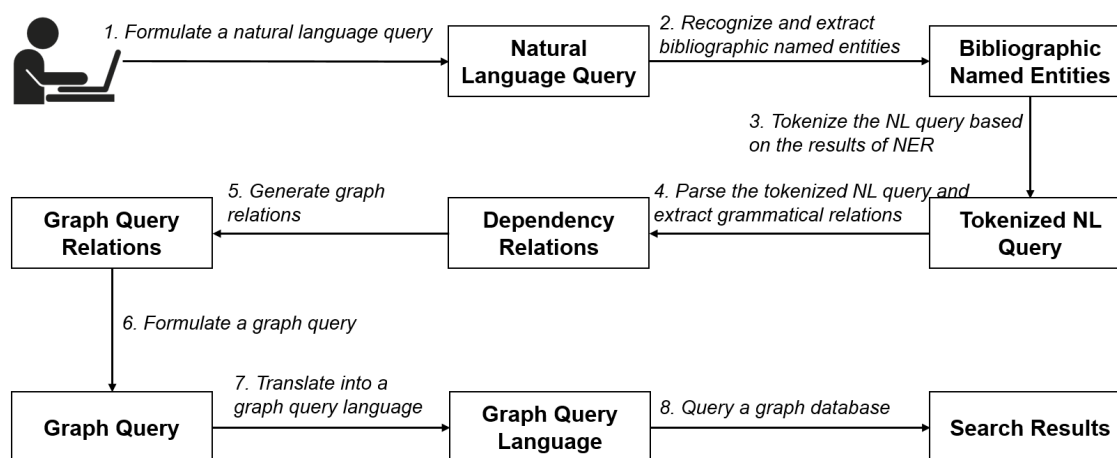


Figure 3.5: The flow chart of the natural language-based system

The steps are as follows: 1) a user formulates a query expressed in natural language; 2) bibliographic named entity recognition is performed by referencing predefined dictionaries and recognized bibliographic named entities are then extracted; 3) a natural language query is tokenized based on the result of bibliographic named entity recognition; 4) the tokenized natural language query is parsed to identify grammatical relations among bibliographic named entities; 5) the grammatical relations are filtered and graph relations are generated ; 6) a graph query is formulated by combining bibliographic named entities and graph relations; 7) the graph query is translated into a graph query language; and 8) a graph database is queried.

3.2.2.2 The Analysis of Natural Language Queries

1) The formulation of natural language queries

Although the system is designed to process a natural language query, it is not a question answering system. Thus, a complete sentence with an interrogative pronoun is not supported in the system. Instead, noun phrases such as “*papers that were written by John*” and “*authors of papers that were published in SIGIR*” are expected queries. Because the interpretation of natural language queries depends on syntactic analysis, queries are expected to have no grammatical error. In addition, relative pronouns, such as “that”, are expected to be included in a query to guarantee that a syntactic parser parses the query correctly. For example, a query “*papers that were written by John*” is the preferred form of “*papers written by John*”.

2) The recognition and extraction of bibliographic named entities

We adopt a dictionary-based named entity recognition approach. We use a simple map structure to construct a dictionary, in which keys are names of bibliographic entities (e.g., “John”, “SIGIR”, and “information retrieval”) and values are their bibliographic types (i.e., Paper, Author, Term, Source, and Organization). These five bibliographic types are regarded as the most useful as shown in previous studies (e.g., Sun, Yu, and Han, 2009). A dictionary is constructed by preprocessing the bibliographic dataset on which we perform searches. Five types of bibliographic instances and their type information are extracted from a self-explanatory dataset. Disambiguation is not performed due to the lack of appropriate identification data. We also add five bibliographic types as keys with annotations to show that they are bibliographic types. For example, the entry <“paper”, “class_Paper”> is added to the dictionary so that the

system recognizes words such as “paper” and “author” in natural language queries. An additional annotation “class_” is added because we want to differentiate five entity types with bibliographic entities.

An approximate string matching algorithm introduced in Gusfield’s work (1997) is used to implement the NER algorithm. In the algorithm, a distance of 1 was assigned to insertion, deletion, and substitution of a character. A maximum distance of 1 was allowed, so that we can recognize plurals or singulars when we have only one form of the two of bibliographic named entities. For example, “Information System” in a query could be identified as a named entity when we only have the term “Information Systems” in our dictionary.

3) The tokenization of natural language queries

We tokenize queries based on the results of named entity recognition to prepare parsing in the next step. After recognizing named entities, we mark named entities of multiples words as single tokens, and then feed queries into a standard tokenizer. This supervised tokenization complements tokenizers’ shortage of domain knowledge on technical terms. For example, without using the results of named entity recognition, terms composed of multiple words such as “information retrieval” will be processed into two different tokens. Tokenization based on the results of named entity recognition can avoid this limitation because terms recognized as a single named entity are treated as one token. Table 3.1 shows the difference between tokenization without NER and with NER using an example query “papers about information retrieval and data mining”, in which tokens are separated by pairs of parentheses.

Table 3.1: Tokenization without NER and with NER

Query	<i>papers about information retrieval and data mining</i>
Tokenization without NER	<i>(papers), (about), (information), (retrieval), (and), (data), (mining)</i>
Tokenization with NER	<i>(papers), (about), (information retrieval), (and), (data mining)</i>

3.2.2.3 The Generation of Graph Queries

1) The parsing of tokenized natural language queries and the extraction of grammatical relations

We use Stanford parser (Klein & Manning, 2003) to parse queries. The output we generate is the Stanford dependencies (De Marneffe et al., 2006) that use 56 grammatical relations to represent binary relations among tokens. Grammatical relations are used to find out which tokens depend on or modify other tokens. For a bibliographic natural language query, parsing is used to find out grammatical relations among bibliographic named entities represented by tokens. Table 3.2 shows the dependency relations of a sample query “*papers about information retrieval and data mining*”. Readers can refer to De Marneffe and colleague’s work (2006) for a detailed explanation of each dependency relation.

Table 3.2: Dependency relations of the query "papers about information retrieval and data mining"

Order	Subject	Object	Relation Code	Relation Name
1		papers	root	root
2	information retrieval	about	case	case marker
3	papers	information retrieval	nmod	nmod_preposition
4	information retrieval	and	cc	coordination
5	papers	data mining	nmod	nmod_preposition
6	information retrieval	data mining	conj	conj_collapsed

For queries that involve citations such as *"papers about information retrieval that were cited by papers that were written by John"*, they are divided into two parts: a cited part and a citing part. By doing so, we reduce the complexities and errors in interpreting queries, because a long list of dependency relations may be error-prone. By dividing the example query into two parts, we no longer need to consider grammatical relations between "papers" in the cited part and "John" in the citing part. This is a practical way to improve the performance of a parser, and thus, words such as "cited", "cites", "cite", and "citing" are used to divide a query into two parts. Parsing is separately applied to each part, and the results are integrated in a later step to generate graph relations.

2) The generation of graph relations from dependency relations

A graph query is a graph representation of a natural language query, in which nodes are recognized bibliographic named entities and links are relations of those entities. Graph relations denote relations that are necessary for building complete graph queries that represent natural language queries. Thus, graph relations are subsets of dependency relations, and graph relations are selected from dependency relations. Irrelevant relations

(i.e., relations among non-bibliographic named entities) that are included in dependency relations are omitted in this process. The selection is performed by considering both the patterns of queries and the database schema that is used to store bibliographic data.

Figure 3.6 shows the algorithm we use to select graph relations from dependency relations. We build the heuristics by combining the test results of a list of expected queries and the database schema. Thus, the heuristics introduced here are dependent on the database schema we use (Figure 3.1) and subject to change if a different schema is employed.

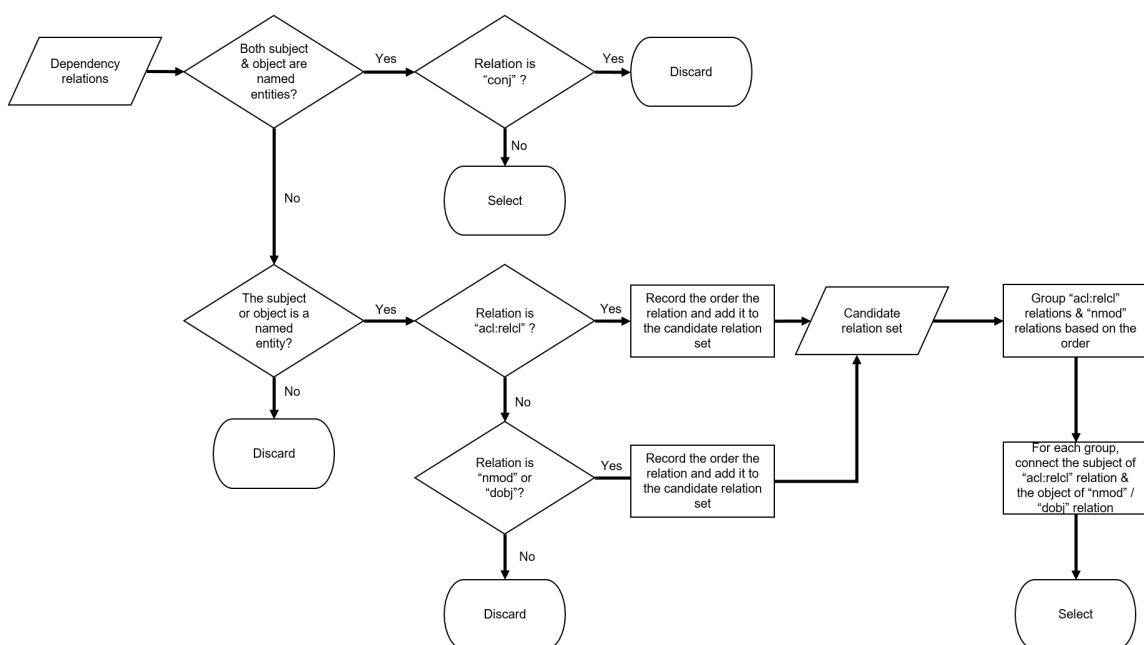


Figure 3.6: The flow chart of selecting graph relations from dependency relations

As shown in Figure 3.6, a relation is selected as a graph relation if both the subject and the object of the relation are named entities. “conj” denotes “conjunct”, and it is used

if two tokens are connected by a coordinating conjunction, such as “and” and “or”. In our case, the relations do not play constructive role in building a graph query, and is thus discarded. Accordingly, the third and fifth relations in Table 3.2 are selected as graph relations while the sixth relation is not. Table 3.3 shows another dependency relations of an example query “*papers that were written by John*”.

Table 3.3: Dependency relations of the query "papers that were written by John"

Order	Subject	Object	Relation Code	Relation Name
1		papers	root	root
2	written	papers	nsubjpass	nominal passive subject
3	papers	that	ref	referent
4	written	were	auxpass	passive auxiliary
5	papers	written	acl:relcl	relative clause modifier
6	John	by	case	case marker
7	written	John	nmod	nmod_preposition

Table 3.3 shows the case in which two bibliographic entities are not directly connected by a dependency relation. It is a normal use case and the algorithm can deal with such use cases. First, the fifth and seventh dependency relations are selected. Then, the subject of fifth relation “papers” and the object of seventh relation “John” are connected to form a new graph relation as shown in Figure 3.6. It is a repeated pattern in bibliographic natural language queries that two relation types “acl”relcl” and “nmod” are used to connect two bibliographic named entities.

3) The conversion of bibliographic named entities to graph nodes

The conversion takes place in three steps. First, we identify the bibliographic named entity that a query is asking. For example, in the query *“papers that were written by John”*, the answer node is “papers”. The identification of an answer node is to locate the object of a “root” relation in parsing results (e.g., “papers” in Table 3.3). Second, we assign each bibliographic named entity a unique instance name that will be used when generating a graph query language. This allows us to differentiate bibliographic named entities with the same name and type the entity “papers” in the query *“papers that were cited by papers that were written by John”*. Lastly, we identify bibliographic named entities that constrain the answer node. For example, “information retrieval” in the query *“papers about information retrieval”* constrains the answer node “papers” by adding a condition. If the type of a bibliographic named entity does not contain the string “class_”, the named entity is a constraint node. This explains the reason that we add the string “class_” to the values of five bibliographic types when constructing the dictionary. Table 3.4 shows instance names, answer nodes, and one or more constraint nodes in the query *“papers that were cited by papers that were written by John”*.

Table 3.4: Graph nodes in the query "papers that were cited by papers that were written by John"

Named Entity	Instance	Answer Node	Constraint Node
papers	cited_Class_Paper_1	Yes	No
papers	citing_Class_Paper_2	No	No
John	citing_Author_3	No	Yes

Information shown in Table 3.4 is an important building block of a graph query language used to query graph databases. It enables the construction of a graph query language by providing all necessary information of nodes in a bibliographic graph.

4) The check of connectedness and directions of graph relations

Connectedness denotes whether two bibliographic named entities are directly connected in a database schema. For example, two bibliographic named entities “papers” and “happy university” in the query “*papers by happy university*” are not directly connected in the schema: “Paper” is connected to “Author” and “Author” is connected to “Organization”. Even though the parsing results suggest a dependency relation between the two bibliographic named entities, the dependency relation should not be selected as a graph relation because it does not conform to the database schema. Thus, we check every dependency relation and add required nodes and relations to form a complete set of graph relations (Figure 3.7).

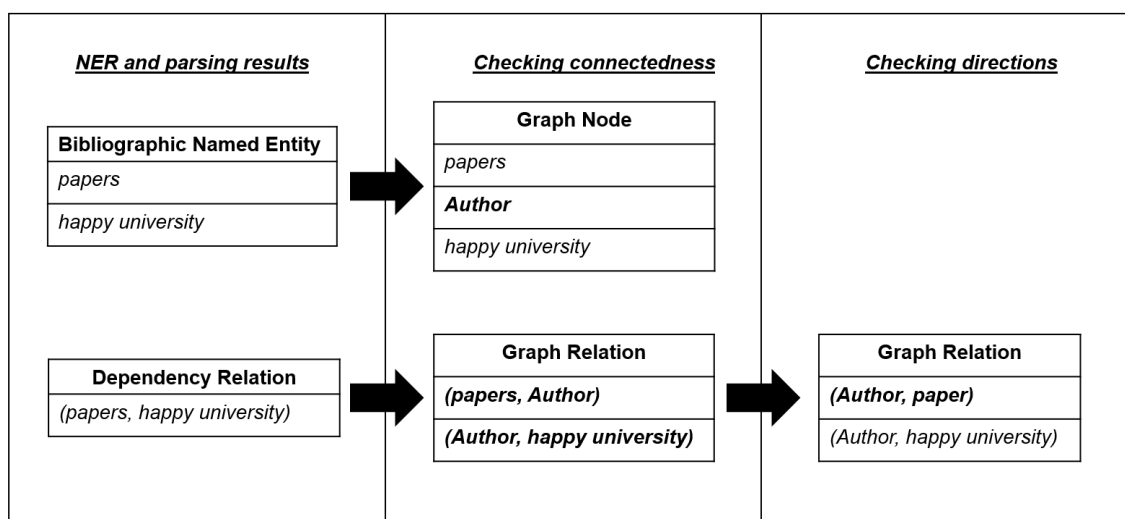


Figure 3.7: The check of connectedness and directions of the query "papers by happy university"

After checking the connectedness of each graph relation and adding necessary new nodes and relations, we check the direction of each graph relation to see whether the source and target of each graph relation conforms to the database schema. In a graph query language, we need to provide a set of graph relations with explicit definitions of sources and targets. For example, the relation between “Paper” and “Author” can be either modeled as “WRITES” or “IS_WRITEEN_BY”, which have different directions. In the above example, the graph relation (papers, Author) was converted into (Author, papers) based on the schema we used.

5) The integration of cited and citing parts

As mentioned previously, we divide a query that involves citations into two parts to reduce the complexities in interpreting natural language queries. These two parts are parsed and converted into graph nodes and graph relations separately. To generate a single graph query, we need to integrate both nodes and relations from two parts. The integration of nodes is achieved by creating a new node set and moving all cited and citing graph nodes to the set. The integration of relations is achieved by connecting two bibliographic named entities with the type of “Paper” in cited and citing parts. If one or two parts do not include a bibliographic named entity with the type of “Paper”, we add a new graph node “Paper” to the part(s) and a graph relation that connects cited paper and citing paper. For example, the query “authors cited by John” denotes authors whose papers that were cited by papers written by John, but both the cited and citing part do not have a bibliographic named entity with the type of “Paper”. Figure 3.8 shows the way to handle such queries.

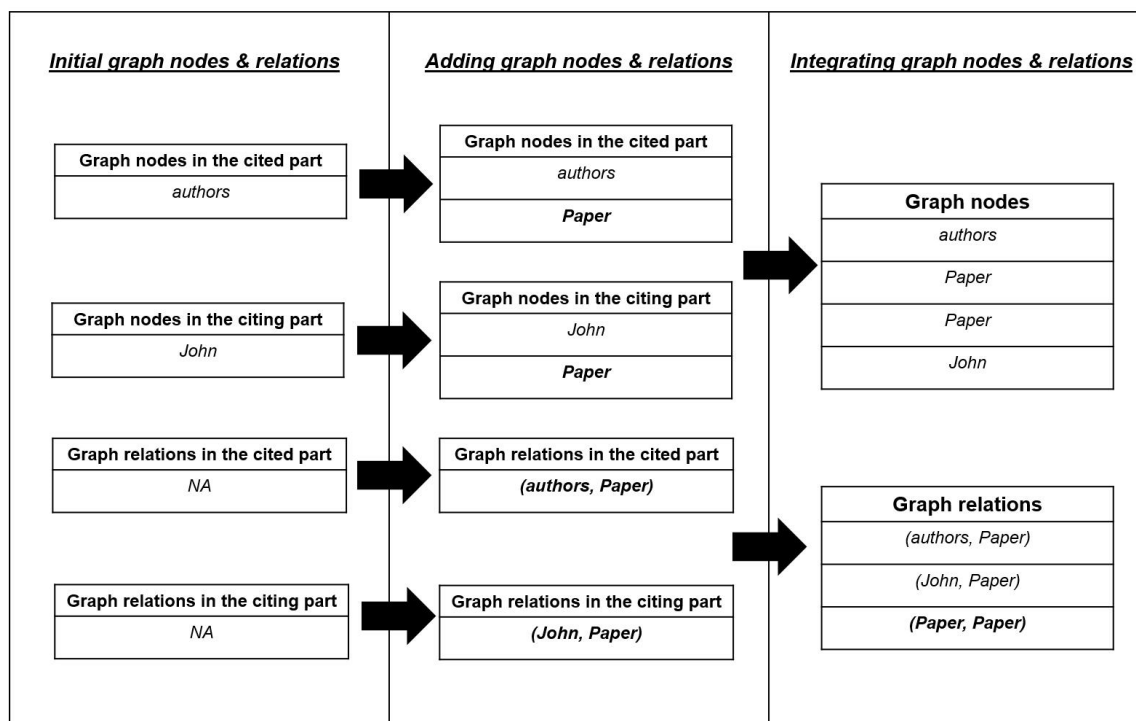


Figure 3.8: The integration of cited and citing parts in the query "authors cited by John"

As shown in Figure 3.8, two “Paper” nodes are added to both cited and citing parts. The nodes are then connected to the existing nodes “authors” and “John”, respectively. Finally, two “Paper” nodes are connected through a citation relation.

3.2.2.4 The Translation of Graph Queries

1) The translation of a graph query into a graph query language

In this step, we translate a graph query into a graph query language. Widely used graph query languages such as Cypher, Germlin, and SPARQL have different syntaxes, but have the same building blocks, i.e., patterns, constraints, and return types. Because graph relations in a graph query are checked for connectedness and directions, and thus

conform to the database schema, they can be directly translated into a graph query language. Constraints and return types are also available as we identify an answer node and constraint nodes in the previous step. Figure 3.9 shows how the graph query of a natural language query “authors that were cited by John” is translated into a graph query language. Four graph nodes derived from four named entities (NE1, NE2, NE3, and NE4) and three relations (R1, R2, and R3) among these graph nodes are identified. These nodes and relation are directly used to generate a graph query language.

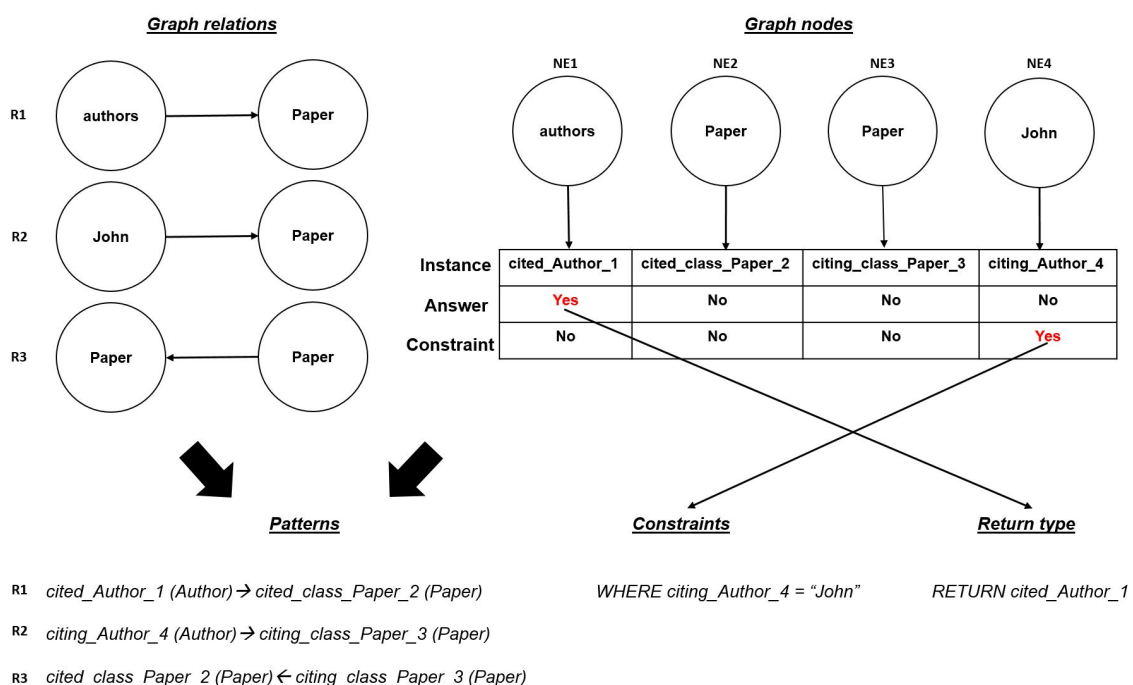


Figure 3.9: The translation of the graph query "authors that were cited by John" into a graph query language

Graph relations are used to derive patterns (i.e., paths), and a constraint is derived from the constraint node (i.e., *citing_Author_4*). The return type in a graph query

language is the answer node (i.e., cited_Author_1) in the graph query. With these three building blocks, a query language can be generated.

2) The query of a graph database

The generated query is submitted to a graph database to retrieve bibliographic data. Another option to query graph databases is to use embedded codes written in programming languages such as Java and C++, as graph databases provide application program interface (API) for data management. However, this approach would reduce the compatibility of a system because graph databases have different APIs. Thus, the framework is designed to translate a natural language query into a graph query language that is supported by a number of graph databases (Holzschuher & Peinl, 2013).

3.3 Visual Graph-based Bibliographic Information Retrieval System

3.3.1 System Overview

Graph queries are a way of searching graph data by taking a graph pattern with a few constraints over nodes and edges as input, which is a natural fit to graph data (He & Singh, 2008). Graph queries are known to convey richer information than other forms of queries and thus improve search performance (e.g., Zhou, Wang, Xiong, Wang, & Yu, 2007). In the visual graph-based system, users formulate bibliographic queries by drawing nodes and their relations. A reference schema is provided so that users can formulate visual graph queries based on it. The system includes a verification module and guides users to formulate a syntactically and semantically correct queries.

3.3.2 System Design

3.3.2.1 Bibliographic Graph Queries

Bibliographic graph queries can be formulated on the basis of the schema (Figure 3.1) with the following additional information:

1) Node type

Every node in a graph query needs to be specified with a type (e.g., Paper). Node type is essential for creating links among nodes because links are created by considering the relations of the types of two nodes. For instance, a node with the type of “Organization” is not linked with a node with the type of “Term”, while it can be linked with a node with the type of “Author”.

2) Answer node

An answer node is the node that answers a visual graph query. For instance, in a visual graph query “*papers on information retrieval that were written by Salton*”, the answer node is “paper”, because the query is asking for returning papers as the final search result. A visual graph query should have at least one answer node. This means that we can retrieve bibliographic entities with more than one type by formulating proper visual graph queries.

3) Constraint node

A constraint node denotes a node that constrains a visual graph query. In the above example query, a node with the name “Salton” is a constraint node that restricts the query to retrieve only papers written by “Salton”. A visual graph query includes one or more constraint nodes.

4) Node name

Unlike answer nodes, constraint nodes should have names in addition to types. Node names are only assigned to constraint nodes because regular nodes do not need names to constrain the query. Figure 3.10 shows four example bibliographic graph queries with varying lengths (i.e., the number of nodes and links), in which the answer nodes are in black and constraints nodes are in red. Regular nodes that connect answer nodes and constraint nodes are in blue. Directions of the links are based on the schema shown in Figure 3.1. As shown in Figure 3.10, bibliographic graph queries are visually represented, and the proposed system is intended to process these visual bibliographic queries drawn by users.

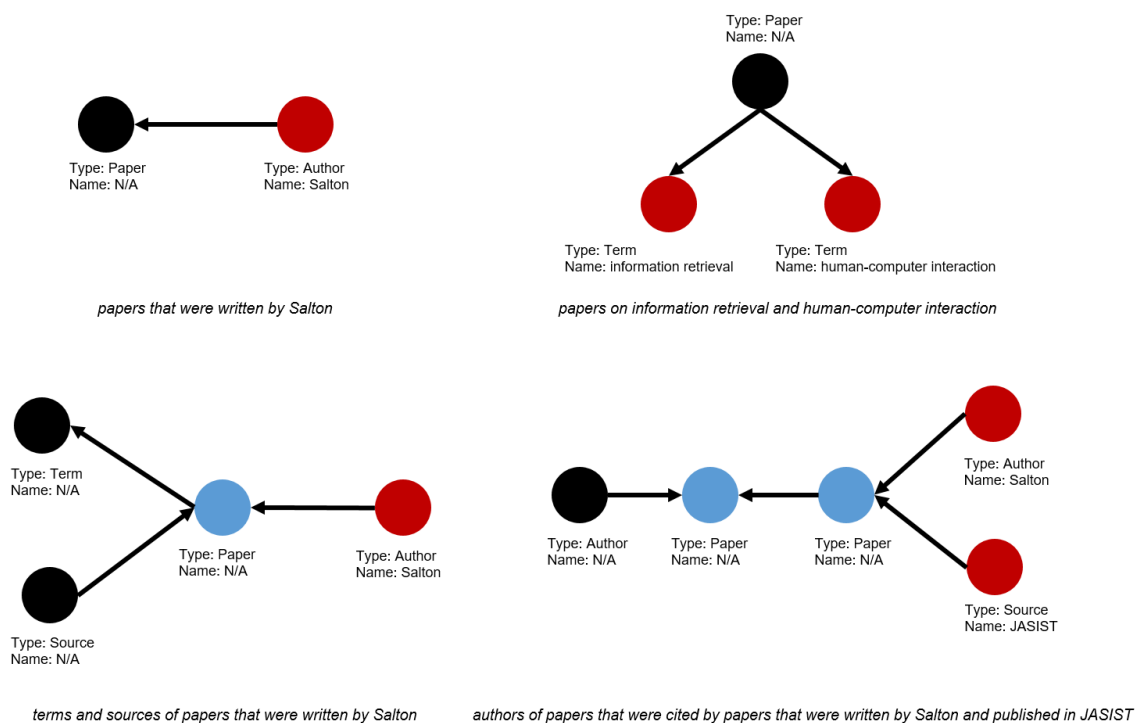


Figure 3.10: Four example visual graph queries

3.3.2.2 System Architecture

The processing of visual graph queries begins with a verification stage because users can create erroneous queries. The verification stage includes both syntax checking and semantics checking. Syntax checking examines whether a visual graph query includes all necessary constructs (i.e., node type, answer node, constraint node, and node name). A syntactically correct graph query is not necessarily a meaningful query, and requires semantics checking. Semantics checking examines whether a visual graph query is answerable by checking the structure of the visual graph query. In this stage, incorrect queries are corrected and ambiguous queries are identified. Possible interpretations of ambiguous queries are sent back to users for their confirmation. Last, a verified visual graph query is translated into a database query to search in a database. Figure 3.11 shows the architecture of a visual graph query-based bibliographic information retrieval system. In the following sections, we discuss in detail the methods involved in the verification of visual graph queries, the generation of candidate graph queries, and the interpretation of graph queries.

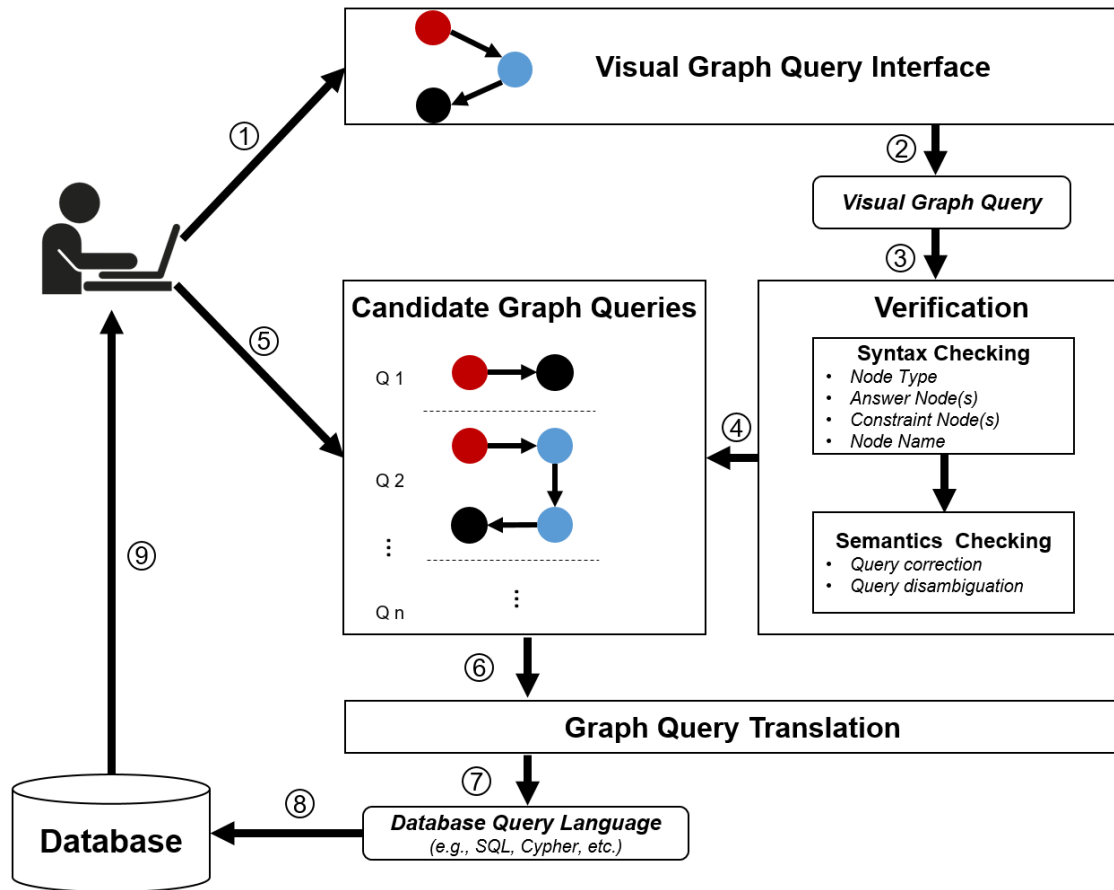


Figure 3.11: Architecture of the visual graph-based bibliographic information retrieval system

3.3.2.3 The Verification of Visual Graph Queries

To satisfy syntax checking requirements, 1) a visual graph query should be a single graph with every single node having at least one relation with other nodes, 2) every node should have a node type, 3) the visual graph query should have at least one answer node, 4) the visual graph query should have at least one constraint node, and 5) all constraint nodes should have names.

In regards to semantic checking, every semantically incorrect link can be divided into three categories (i.e., shortest path equals to zero, shortest path equals to one, and shortest path greater than one) based on the length of the shortest path between the source and the target of the link. Shortest paths are obtained from the data schema of bibliographic data (Figure 3.1). We apply the following algorithm (Figure 3.12) to a visual graph query for semantics checking. We perform query correction and query disambiguation for syntactically incorrect queries. We check every link of a visual graph query and perform query correction and disambiguation on the basis of links.

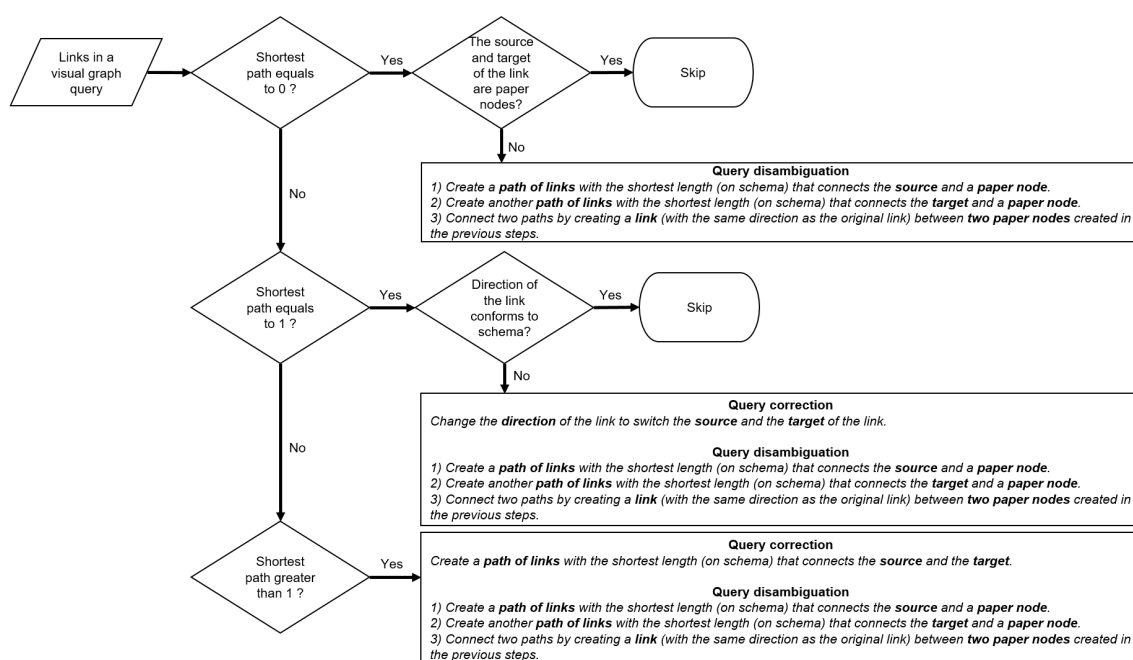


Figure 3.12: An algorithm for query semantics checking

As shown in Figure 3.12, every link of a visual graph query is checked for the length of shortest path between the source and the target. Any link that does not conform to the

schema is then updated. Query correction is performed when the direction of a link is incorrect (e.g., from a paper node to an author node) or two nodes are connected when they are not directly related on the schema (e.g., a link from a paper node to an organization node). While query correction is achieved for two of three categories (i.e., shortest path equals to one and shortest path greater than one); query disambiguation is accomplished for all three categories to generate all possible candidate graph queries.

Figure 3.13 shows three examples of semantically incorrect links as well as how they are corrected and disambiguated. Letters inside the nodes denotes node types (i.e., A for author, P for paper, T for term, S for source, and O for organization). Names of constraint nodes (red nodes) are shown under the table.

Graph Relation	Shortest Path	Query Correction	Query Disambiguation
	0	N/A	
	1		
	>1		

A • "John"
S • "SIGIR"

Figure 3.13: Examples of query correction and disambiguation

If the source and the target of a link are the same, we treat the shortest path between the source and the target as zero. In the first example, both the source and the target is Author. This is semantically incorrect because there is no link between the two author nodes based on the schema. Even though the first example is incorrect in semantics, it could be formulated by users with a specific meaning, i.e., authors cited by John, which should be authors who wrote papers that were cited by papers written by John. Thus, for a link with shortest path equals to zero, we disambiguate the link by treating the link as a citation relation and adding two additional Paper nodes as shown in the last column of the first example. Two Paper nodes are connected with the source and target respectively by finding their shortest paths from the source node and to the target node. Finally, two paper nodes are connected with a link that has the same direction as the original link because the direction of the original link is considered as the direction of the citation relation. For the second example, the direction of the link is opposite to that of the schema: in the schema, the link between a Paper and an Author is from an Author to a Paper with the link label of WRITES. Thus, we correct the link by changing the direction of the link. It is also possible that the user may mean papers cited John (i.e., papers cited papers that were written by John). Thus, query disambiguation is performed by creating a new graph query with the above meaning. Similarly, the third example can either mean authors that presented papers in SIGIR or authors who wrote papers that cite papers that were presented in SIGIR. Query correction and disambiguation are performed, and two new paths are created. In this way, query correction and disambiguation helps users formulate correct visual graph queries.

3.3.2.4 The Generation and Interpretation of Graph Queries

1) The generation of candidate graph queries

Candidate graph queries are the enriched version of visual graph queries, in which links are added with labels. Candidate graph queries are generated as the results of the verification of visual graph queries. If a user formulates a syntactically and semantically correct visual graph query, there would be only one candidate graph query. Otherwise, there would be more than one candidate graph query. As shown in Figure 3.13, a semantically incorrect link resulted in one or two paths/links through the process of semantics checking. Because a visual graph query comprises one or more links, if a visual graph query includes two semantically incorrect links, there would be up to four possible interpretations. If a query includes n incorrect links, the maximum number of candidate graph queries would be $2n$. Given a visual graph query, we generate all possible candidate graph queries and return them back to the user who formulated the visual graph query. The user then selects one that best represents his/her information needs to proceed. Figure 3.14 shows a visual graph query with two semantically incorrect links (dotted links) and candidate graph queries generated from the visual graph query.

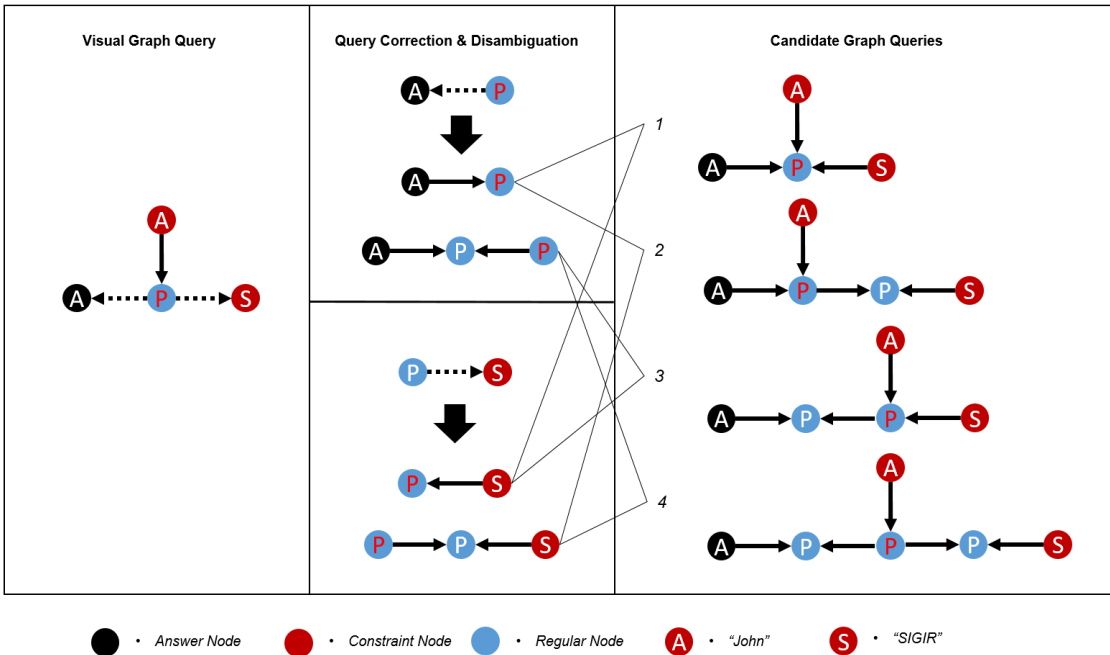





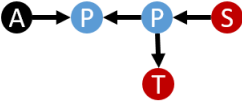
Figure 3.14: An example of generating candidate graph queries

As shown in Figure 3.14, each semantically incorrect link resulted in two possible interpretations by ways of query correction and disambiguation. We combined all possible interpretations and generated four candidate graph queries. The user can select one to proceed. The next step is to translate the selected graph query into a database query.

2) The translation of graph queries into Cypher commands

We show how graph queries can be translated into Cypher commands that can query data stored in Neo4j (a graph database), which is implemented based on the property graph model shown in Figure 3.1. Figure 3.15 shows four example graph queries and

their translations into Cypher commands. The translation of graph queries to Cypher commands is simple, because Cypher is itself a query language based on graph patterns. Other graph database query languages such as SPARQL and Gremlin have similar syntaxes with Cypher, and the general constructs are primarily the same. Readers can refer to Holzschuher and Peinl's work (2013) for a comparison of different graph query languages.

Visual Graph Query	MATCH	WHERE	RETURN
	<code>(a:Author)-->(p:Paper)</code>	<code>a.name = 'John'</code>	<code>p.name</code>
	<code>(a:Author)-->(p:Paper), (p:Paper)-->(s:Source)</code>	<code>a.name = 'John'</code>	<code>s.name</code>
	<code>(o:Organization)-->(a:Author), (a:Author)-->(cited_p:Paper), (cited_p:Paper)-->(citing_p:Paper)</code>	<code>citing_p.name = 'Introduction to Database Systems'</code>	<code>o.name</code>
	<code>(a:Author)-->(cited_p:Paper), (cited_p:Paper)-->(citing_p:Paper), (citing_p:Paper)-->(s:Source), (citing_p:Paper)-->(t:Term)</code>	<code>t.name = "Information Retrieval" AND s.name = "SIGIR"</code>	<code>a.name</code>








	• Answer Node		• "John"		• "Introduction to Database Systems"
	• Constraint Node		• "SIGIR"		• "Information Retrieval"
	• Regular Node				

Figure 3.15: Examples of translating visual graph queries into Cypher

The three main constructs of Cypher are MATCH, WHERE, and RETURN. We specify graph patterns in a MATCH clause and set constraints to nodes in a WHERE clause. A RETURN clause is for the data that we want to retrieve. A RETURN clause in

Cypher corresponds to a SELECT clause in SQL, and both are for specifying data need to be returned by the database. The WHERE clause in Cypher is much simpler than the one in SQL. We only need to set values to the bibliographic entities that are designated as constraint nodes when a user formulating a graph query. Constructing a MATCH clause is straightforward, which is a textual representation of a graph query, with the same nodes and relations with the same directions. The only thing we need to do is to set an arbitrary and unique instance name for each node. Instance names can be simply strings with/without numbers that can differentiate nodes from each other. An instance name of a node (e.g., cited p) is accompanied by the node type (e.g., Paper), which guides the traversal of the graph.

4. Results

This section discusses system use cases and reports results on system-centered experiments. We show how to use the systems through example queries and present experimental results on advanced features of the systems.

4.1 Form-based Bibliographic Information Retrieval System

4.1.1 A System Use Case

In this section, we show an example of our system and test various features. The sample data set (Figure 3.3) comprises five articles (in blue), five authors (in red), seven terms (in purple), three sources (in green), and three affiliations (in yellow). It shows 23 entities and their relations. Links between a pair of papers (i.e., citation relations) have directions whereas other links do not.

The example natural language query is *“authors who are affiliated with Happy University and wrote papers that were cited by papers on NoSQL published in VLDB.”* As shown in Figure 3.4, we specified citation type as “cited” because the target we want to search is at the cited side. Then, we added anchor types to both sides. Because the authors are affiliated with “Happy University,” we added an anchor type—Organization at the cited side and put the value—“Happy University.” Similarly, we added two anchor types—term and source to the citing side, and put values—“NoSQL” and “VLDB.”

Then, by clicking “Generate Graph Query,” the system generated a graph query which is shown in Figure 3.4. Black node is the target—author, red nodes are the anchors— Organization, Term, and Source. Figure 4.1 shows the Cypher query generated by the system.

```
MATCH (cited_o:Affiliation)<--(cited_a:Author),
      (cited_a:Author)-->(cited_p:Paper),
      (cited_p:Paper)<--(cited_s:Source),
      (cited_p:Paper)-->(cited_t:Term),
      (citing_o:Affiliation)<--(citing_a:Author),
      (citing_a:Author)-->(citing_p:Paper),
      (citing_p:Paper)<--(citing_s:Source),
      (citing_p:Paper)-->(citing_t:Term),
      (cited_p:Paper)<--(citing_p:Paper)
WHERE cited_o.name= 'Happy University' AND
      citing_t.name= 'NoSQL' AND
      citing_s.name= 'VLDB'
RETURN DISTINCT cited_a.name
```

Figure 4.1: Generated Cypher query

The last step is to translate form queries into Cypher and return search results. The search results included two authors— “Linda” and “Mary,” which are relevant answers based on the information provided in Figure 3.3.

4.1.2 Experiments

In this section, we evaluate the proposed system in two aspects: (a) we show the functional limitations of the current bibliographic information retrieval systems to highlight the functionalities of the proposed system; (b) we compare two different database models to highlight the performance of the graph data model.

4.1.2.1 Functional Limitations of Current Bibliographic Information Retrieval Systems

We chose the Web of Science as a representative system to show two functional limitations of the current bibliographic information retrieval systems: a limited support of entity types as the final search results and a limited support of complex queries. As previously mentioned, current bibliographic information retrieval systems only provide articles as the final search results. Thus, these systems cannot appropriately answer a query like “*terms of papers that were written by author A (e.g., Lutz Bornmann) and published in journal A (e.g., JASIST).*” This example query might be useful to identify important terms/keywords of an author’s seminar research given that journals in the query codify seminar research. In order to answer this query, one can first retrieve an author’s papers that were published in JASIST by designating “Publication Name” as the “Journal of the Association for Information Science and Technology” and set the “Author” field as, for instance, “Bornmann L.” Nonetheless, we cannot retrieve terms because the search results only contain papers. Even though some metadata (e.g., Research Areas and Organizations) are available as categories for refining the search results, these metadata just serve to support the retrieval of articles. On the other hand, the proposed system can

answer the example query as shown in Figure 4.2. Taking various entity types into consideration and enabling the search for these entities is a desired function of modern bibliographic information retrieval systems.

The screenshot displays the GIBIR web application interface. The browser window title is "GIBIR" and the address bar shows "localhost:8090/gir/". The main heading is "GIBIR: A Graph-based Interactive Bibliographic Information Retrieval System". The interface is divided into three sections: "Form Query", "Graph Query", and "Search Results".

Form Query: This section contains a "Citation" dropdown menu with options "N/A", "Cited", and "Citing". Below it is a "Target:" dropdown menu set to "Term". The "Anchor:" section includes an "Add an Anchor" button and two input fields. The first field is labeled "Author" and contains the text "Name: Robert". The second field is labeled "Source" and contains the text "Name: JASIST". There are "Generate Graph Query" and "SEARCH" buttons at the bottom of this section.

Graph Query: This section displays a graph with five nodes: "Term" (black), "Author (Robert)" (red), "Source (JASIST)" (red), "Paper" (blue), and "Affiliation" (blue). Edges connect "Term" to "Paper", "Author (Robert)" to "Paper", "Source (JASIST)" to "Paper", and "Author (Robert)" to "Affiliation".

Search Results: This section shows "Total: 2 results." and a list of results including "Bibliometrics" and "IR".

Figure 4.2: The search results for the example query

The Web of Science provides a way of representing complex queries under “Advanced Search” through the use of filed tags and Boolean operators, but this is limited to queries without citation. While the Web of Science also supports “Cited Reference Search,” which is a function of retrieving citing articles given a cited article or author, one cannot search for cited articles under this function. Thus, there is the need to support representing complex queries by integrating advanced search, citing article

search, and cited article search. To address this, the proposed system supports the composition of complex queries and search for both cited and citing articles—both citing and cited entities that include authors, affiliations, papers, sources, and terms can be retrieved through a one-step operation (i.e., by filling forms and querying databases). The proposed system thus enables a search such as “*papers on information retrieval, which were cited by John’s papers that had been presented in SIGIR*” and “authors or sources that cited a particular JASIST paper.”

4.1.2.2 A Comparison between Graph Data Model and Relational Data Model

In order to test the performance of the proposed system, in particular, the use of the graph database, we constructed another system which has the same interface and features, but uses a relational database as the underlying infrastructure. The two systems have the same query generation and refinement components with the only difference being the data storage layer. Thus, in this section, we compare the performance of a graph database and a relational database. Because a relational database uses a relational data model, a completely different conceptual model is needed. Figure 4.3 shows a conceptual model (i.e., ER diagram) for the relational database. This conceptual model was later physically implemented in the MySQL database.

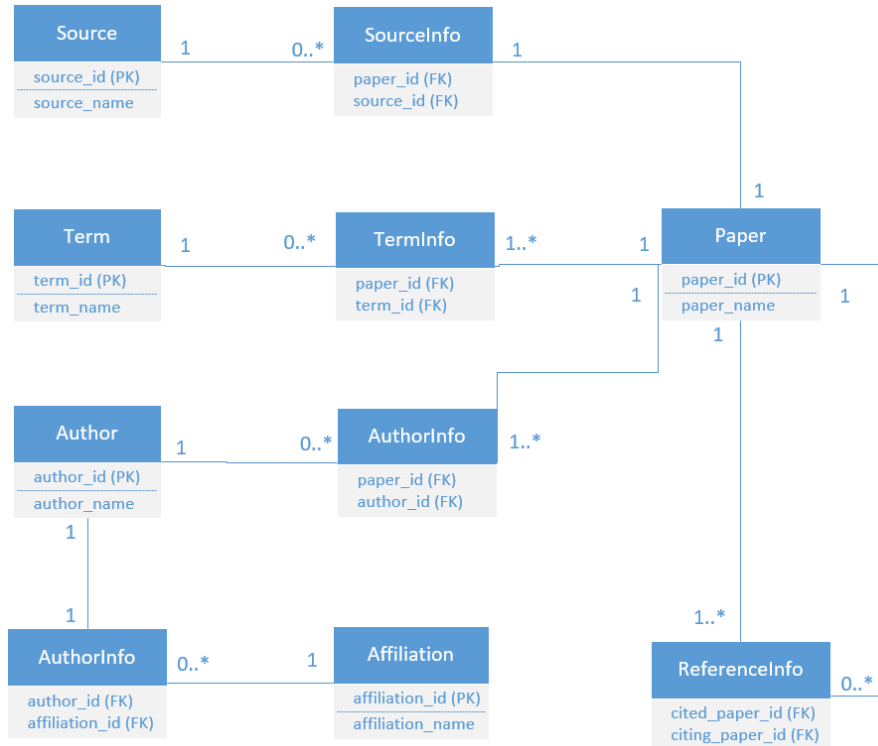


Figure 4.3: An ER diagram for the relational database-based system

As for the experiment data set, we used a data set provided by Tang et al. (2008). This data set includes information on papers, titles, authors, publication venues, abstracts, and cited references. The original data set does not contain information on terms and organizations. We randomly assigned 1,000 universities to authors, that is, from “University1” to “University1000” for each author. We tokenized titles of papers based on space, and considered those tokens as key terms of papers. Some papers have references that are not included in the data set, and we removed these references in order to maintain consistency. This text-based data set was parsed and populated into Neo4j and MySQL by following their respective schemas. Table 4.1 shows the number of each bibliographic entity used in this study.

Table 4.1: The number of bibliographic entities and relations in the dataset

Entities	Count	Relations	Count
Paper	629,814	Paper-Paper	632,751
Author	595,775	Paper-Author	1,312,057
Source	12,609	Paper-Source	531,219
Term	291,109	Paper-Term	5,270,539
Affiliation	1,000	Author-Affiliation	595,775

A few queries were constructed for the experiment. Queries can be divided into several groups based on the number of nodes (i.e., bibliographic entities) in a query. Queries involving many nodes are inherently more complex than queries that include only a few nodes. We constructed four groups of queries, that is, from a query with two nodes to a query with five nodes. Queries made up of four or five nodes are the ones with citation relations. Each group includes a few meta-paths. We selected one meta-path for each group by considering their universality—we selected meta-paths that seemed to appear frequently in search cases. Table 4.2 shows selected meta-paths and examples of natural language queries.

Table 4.2: Meta-paths and example natural language queries

Nodes	Meta-path	Example Natural Language Query
2	Paper-Author	Papers written by “ <i>Author</i> ”.
3	Source-Paper-Author	Sources (i.e., conferences/journals) published papers written by “ <i>Author</i> ”.
4	Affiliation-Author-Paper1-Paper2	Affiliations of authors of papers cite/cited by “ <i>Paper</i> ”.
5	Affiliation-Author-Paper1-Paper2-Term	Affiliation of authors of papers cite/cited by papers about “ <i>Term</i> ”.

Because the two systems use the same interface, we tested the query execution time which is the duration between the point of sending queries and that of finishing retrieving items. Given the characteristic of metadata search like the proposed system, the results are always correct as long as we provide correct information. Query execution time is an optimal measure because we can leave out human factors (e.g., the time a user spends generating queries) and focus on system-intrinsic elements given that the two systems have the same query generation and refinement interfaces. For each meta-path listed in Table 3.2, we constructed 10 queries and recorded the query execution time at the system level by the embedded time checker. The test environment is a desktop PC with a Windows 7, 64-bit operating system, an Intel Core i7-3770 CPU, and 20GB RAM. Figure 4.4 shows the query execution time for the tested queries in both systems.

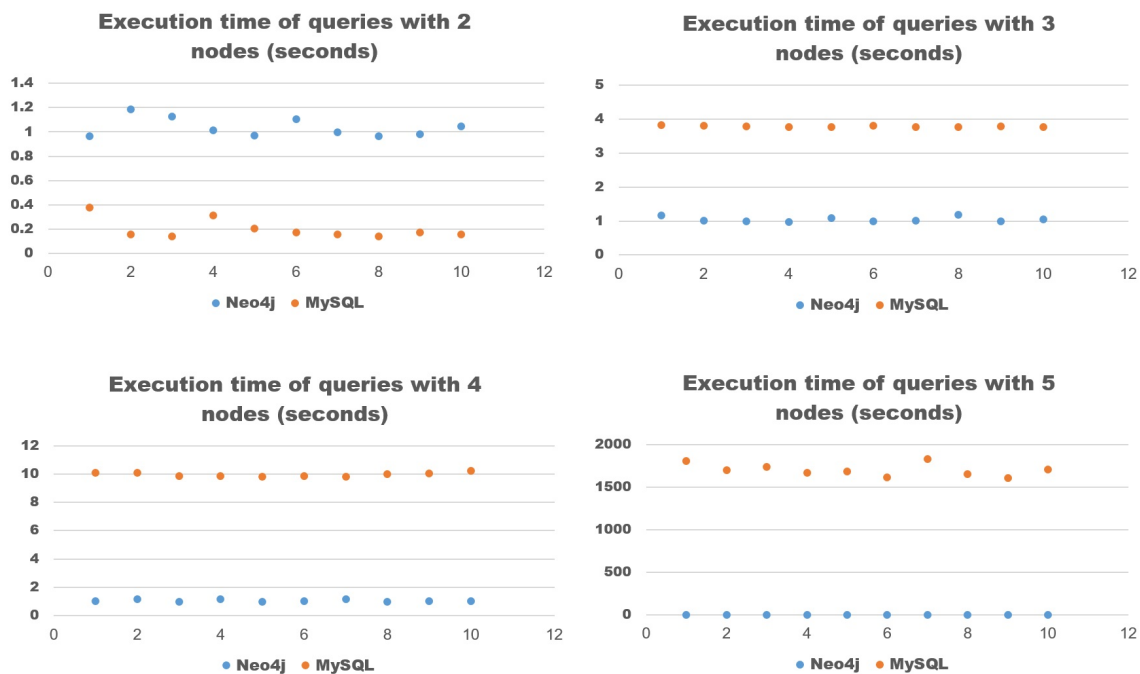


Figure 4.4: Query execution time of queries of different groups

As shown in Figure 4.4, the relational database (i.e., MySQL) performed better in executing queries with two nodes. The relational database executed all queries in less than 0.4 second while the graph database spent about 1 second to process most queries. However, as queries became more complex, the graph database (i.e., Neo4j) outperformed the relational one. The execution time of the relational database increased from less than 4 seconds for queries with three nodes to about 30 minutes for five-node queries. One reason that affected the execution time is the join operations used in relational databases. Join is a SQL operation used in relational databases to combine records from two or more tables to get final records (Mishra & Eich, 1992). As queries involve more nodes, more join operations are needed. In terms of processing a query with five nodes, the relational database processed five join operations. Because a table has

many records (e.g., the paper table has 629,814 records), processing these operations is very time-consuming. Another reason is attributed to the time used to retrieve indexed items. Relational databases use indices to presort data in order to facilitate fast retrieval (Lahdenmaki & Leach, 2005). Although indices are useful to locate data quickly, we still need additional time to traverse the indexed data to discover their relations. This is because relations between two records are not explicit, and traversal is needed to find out the existing connections.

In graph databases, joins and traversals (in terms of finding relations) are not an issue. Graph databases are not made up of tables; thus, graph databases do not require time-consuming join operations. For this reason, the query execution time of different query groups in the graph database was not affected by the number of nodes in queries. As a result, the query execution times of different query groups are practically the same (Figure 4.5) while the performance of the relational database varies significantly. In addition, graph databases have the property of “index-free adjacency” (Robinson, Webber, & Eifrem, 2013). In graph databases, traversals among relations are not required because a node physically keeps information of the connected nodes. This means as long as we locate a node, other nodes that have relations with this node are immediately shown, and no other effort is needed to locate relations. Thus, graph databases benefits from traversals and this benefit becomes more apparent as the number of relations between records increases.

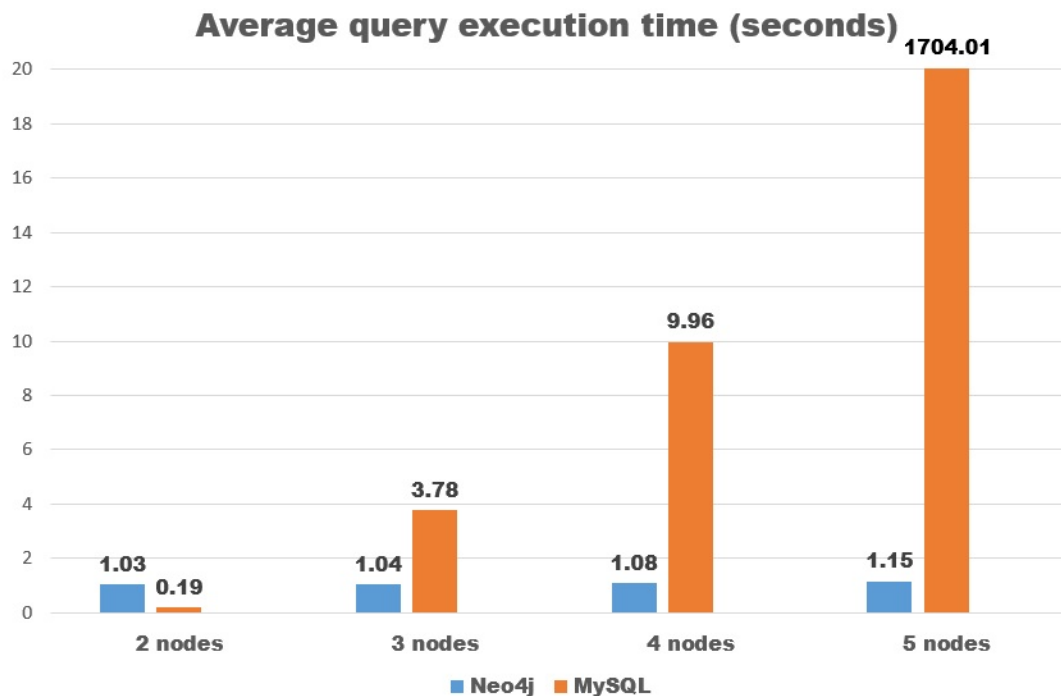


Figure 4.5: Average query execution time of queries

The experiment results showed that the relational database-based system spent more than 30 minutes in processing five-node queries. In reality, a system with such a performance would not be practical. This could be partly ameliorated by dividing a query into several short queries and refining search results accordingly—an approach practiced by current bibliographic information retrieval systems such as the Web of Science. Thus, the comparison of the execution time of five-node queries highlights the applicability of the proposed graph database-based bibliographic information retrieval system.

4.2 Natural Language-based Bibliographic Information Retrieval System

4.2.1 A System Use Case

Figure 4.6 shows the graphical interface for users to formulate natural language queries. The example query is “*Papers about classification, which were cited by Asoke K. Nandi 's papers that had been presented in Pattern Recognition*”.

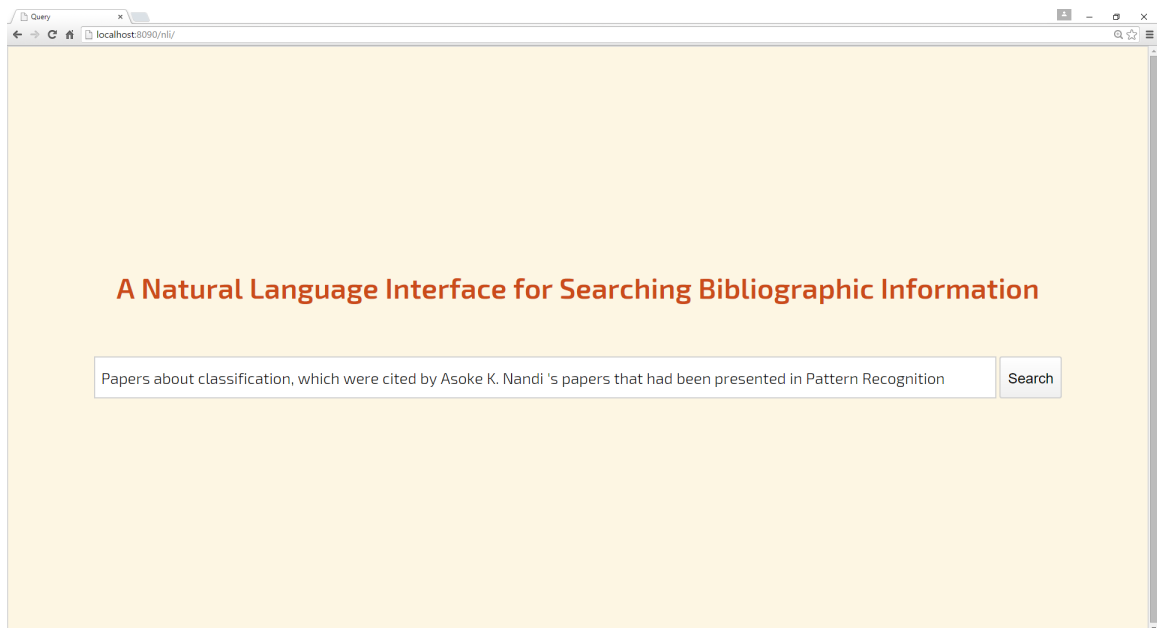


Figure 4.6: A natural language interface with an example query

After typing the natural language query and clicking the “Search” button, the system analyzes the natural language query. Recognized bibliographic named entities, dependency relations of the query, graph nodes, graph relations, and a graph query are shown in Figure 4.7.

Search
localhost8090/nli/search
Papers about classification, which were cited by Asoke K. Nandi 's papers that had been presented in Pattern Recognition

Named Entity Recognition & Parsing

Named Entities

Token Position	Name	Type
0	Papers	Paper
2	classification	Term
8	Asoke K. Nandi	Author
16	Pattern Recognition	Source

Dependencies

Subject	Object	Relation (Abbreviation)	Relation (Full Name)	Citation
Papers	root	root	root	cited
classification	about	case	case marker	cited
Papers	classification	rmod	rmod_preposition	cited
were	classification	rsubj	nominal subject	cited
classification	which	ref	referent	cited
classification	were	ecrelcl	relative clause modifier	cited
papers	Asoke K. Nandi	rmod_poss	possession modifier	citing
Asoke K. Nandi	's	case	case marker	citing
papers	root	root	root	citing
presented	papers	rsubppass	nominal passive subject	citing
papers	that	ref	referent	citing
presented	had	aux	auxiliary	citing
presented	been	auxpass	passive auxiliary	citing
papers	presented	ecrelcl	relative clause modifier	citing
Pattern Recognition	in	case	case marker	citing
presented	Pattern Recognition	rmod	rmod_preposition	citing

Graph Query

Graph Query Nodes

Node Name	Node Type	Node Instance	Constraint	Answer
Papers	Paper	cited_class_Paper_#cde	true	true
classification	Term	cited_Term_3	true	false
Asoke K. Nandi	Author	citing_Author_3	true	false
Pattern Recognition	Source	citing_Source_4	true	false
paper	Paper	citing_Paper_5	false	false

Graph Query Relations

Source	Target
Papers	classification
Asoke K. Nandi	paper
Pattern Recognition	Papers

Results

Figure 4.7: The analysis of a natural language query

Figure 4.7 shows how the example query was analyzed. First, bibliographic named entities such as Papers, classification, Asoke K. Nandi, papers, and Pattern Recognition were recognized. As mentioned previously, these bibliographic named entities were extracted from the dataset we used in the experiment and stored into a dictionary. Dependency relations among all tokens in the query are also shown as the result of a syntactic analysis. Nodes were then obtained from bibliographic named entities while relations were selected from dependency relations. By integrating graph nodes and graph relations, the system generated a graph query to visualize the results of the natural language query. As an interactive information retrieval system, users can modify or proceed with the current natural language query by referencing the analysis of the graph query. The final search results are obtained by clicking the “Results” button (Figure 4.8).

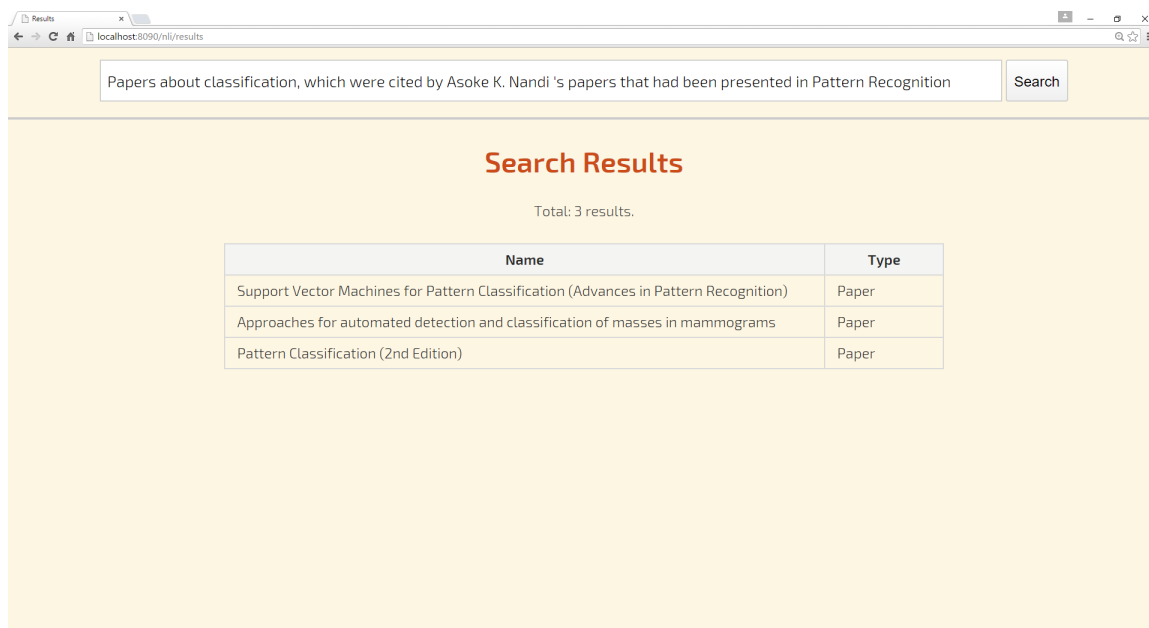


Figure 4.8: The search results of the example query

Figure 4.8 shows the final search results, which are the correct answers for the example query. In the system, we used Cypher as the graph query language, which is the default query language of Neo4j. Search results showed that there are three entries that matched the natural language query.

4.2.2 Experiments

The effectiveness of the system is evaluated as the ratio of correctly answered queries and query execution time, as practiced in related research (e.g., Li & Jagadish, 2014; Tablan et al., 2008). We tested both the ratio of correctly answered queries and query execution time by forming four groups of queries based on the number of bibliographic named entities in a query, which ranges from two-named entities to five-named entities.

Ten queries for each group were tested. When formulating test queries, we considered a variety of meta-paths and included as many meta-paths as possible. For example, for two-node queries, we included meta-paths such as “Author→Paper”, “Author→Organization”, “Source→Paper”, “Paper→Term”, and “Paper→Paper”. As the number of named entities in a query increases, the number of meta-paths also grows. Therefore, we selected 10 meta-paths that are representative in bibliographic searching based on our domain knowledge. Forty tested queries are listed in the Appendix A. The ratio of correctly answered queries for each group is shown in Table 4.3.

Table 4.3: The ratio of correctly answered queries

The number of named entities	2	3	4	5
The ratio of correctly answered queries	10/10	10/10	9/10	10/10

As shown in Table 4.3, we did not see a correlation between the number of named entities in a query and the ratio of correctly answered queries. The example query that our framework processed incorrectly is “*Authors who are affiliated with University007 and wrote Papers about clustering*”. The reason of the misinterpretation is that the parser misidentified “wrote” as the root of the query, which should be “authors”. Our system performed 100% correctly for all other test queries.

Query execution time includes the time of interpreting a natural language query (i.e., recognizing named entities and parsing) and the time of answering the query in a graph database. Time spent in formulating a query is not considered to leave out human factors and to focus on the performance of the system. The test environment is a laptop PC with

a Windows 7 64-bit operating system, an Intel Core i5-3320M CPU, and 16GB RAM. The execution time for each query and average execution time in each group are shown in Figure 4.9. The query that was incorrectly interpreted was excluded from the calculation.

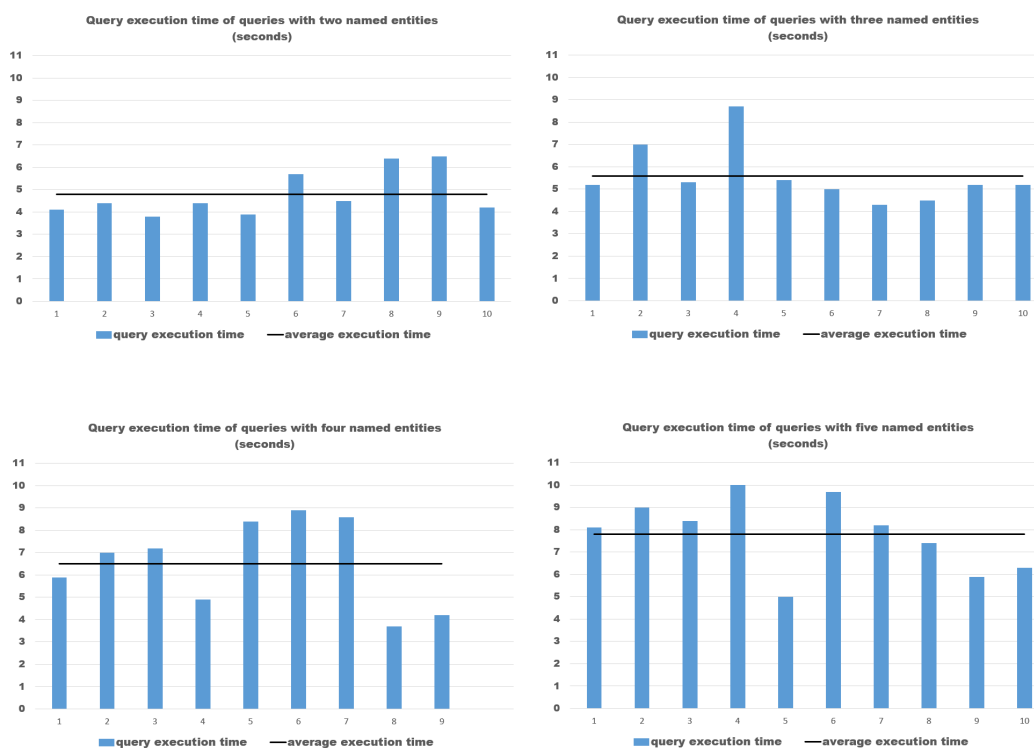


Figure 4.9: The query execution time of queries with the number of named entities from two to five

Query execution time is affected by the length of a query as well as the number of items in the search result that matched the query. A long natural language query need more time to be interpreted than a short query as the time spent on recognizing named entities in the query and parsing the query increases. Query execution time also increases if there are many items that matched the query. The average execution time is 4.8

seconds for two-named entity query, 5.6 seconds for three-named entity query, 6.5 seconds for four-named entity query, and 7.8 seconds for five-named entity query. The longest time taken to process a query is about ten seconds. Nonetheless, an industry-scale systems use more powerful servers, we believe the execution time should be reduced in real-world use cases.

4.3 Visual Graph-based Bibliographic Information Retrieval System

4.3.1 A System Use Case

Figure 4.10 shows the visual graph-based system. The visual graph query interface is composed of two panels: the configuration panel (left) and the graph query panel(right). The configuration panel has three sections: a node configuration section, a schema section, and an instructions section. As shown in the instructions section, users can create a node, drag between nodes to add a link, and change the direction of the link by pressing the Z key. Deleting a node or a relation can be done by pressing the Delete key after selecting a node or a relation. The visual graph query shown in Figure 4.10 has five nodes and each node has an id from zero to four. Attributes are shown next to the nodes; for example, node 1 has a string value of “Type: Author | ANSWER: Y | CONSTRAINT: N | Name: N/A”. It means that the type of node 1 is Author, and it is an answer node. In addition, the node is not a constraint node and does not have a name. Users can select a node (e.g., node 4) to set and change attributes of the node. Node 4 is a constraint node and it has the name of “Communications of the ACM”. The visual graph query shown in

Figure 4.10 denotes “*authors of papers that were cited by papers that were written by Gerard Salton and published in Communications of the ACM*”.

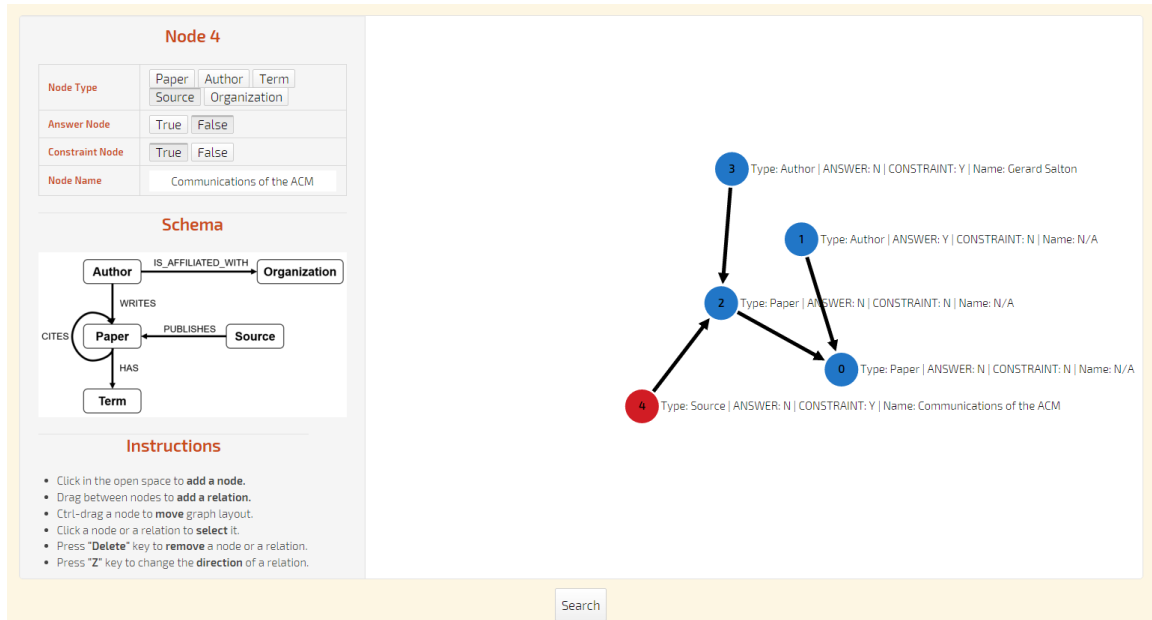


Figure 4.10: The visual graph interface of the system

By clicking the search button, candidate graph queries are shown (Figure. 4.11(a)). Only one candidate graph query is shown because the formulated graph query is syntactically and semantically correct. As shown in Figure 4.11, all links are added with labels and each node is filled with appropriate colors (i.e., black for the answer node, red for constraint nodes, and blue for regular nodes). Search results of the candidate graph query is shown in Figure 4.11(b).

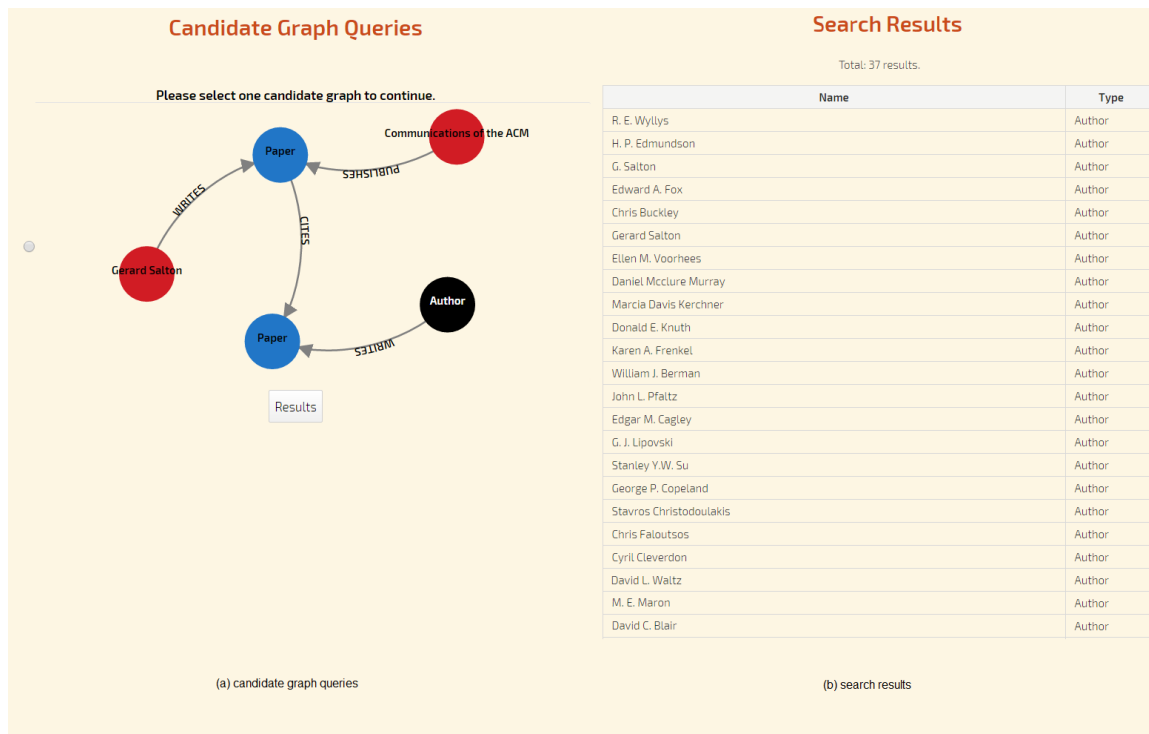


Figure 4.11: Candidate graph queries and search results

Next, we show how a semantically incorrect visual graph query is corrected and disambiguated. Figure 4.12 shows an example query, in which the link from node 0 to node 2 is incorrect. The direction of the link is opposite to the schema, and thus needs to be further analyzed to identify all possible interpretations.

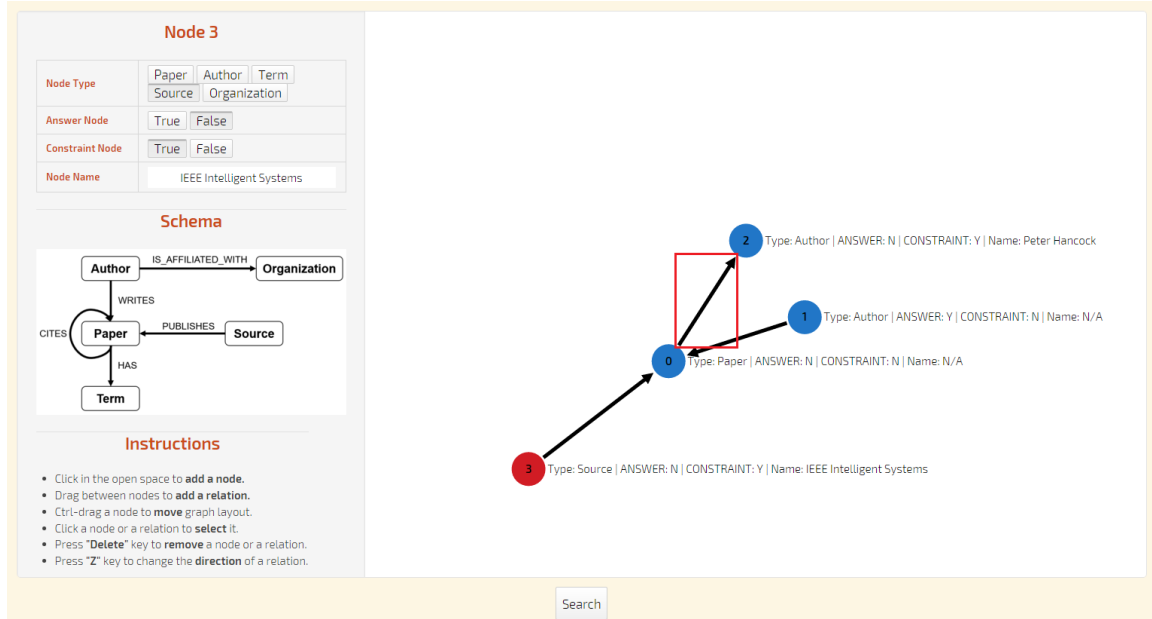


Figure 4.12: A semantically incorrect visual graph query

As shown in Figure 4.13, two candidate graph queries are formulated from the above visual graph query for users to select. The visual graph query can be interpreted as either “*authors of papers that were written by Peter Hancock and published in IEEE Intelligent Systems*” (for finding co-authors of Peter Hancock) or “*authors of papers that were published in IEEE Intelligent Systems and cited papers that were written by Peter Hancock*”. The two candidate graph queries resulted in different search results.

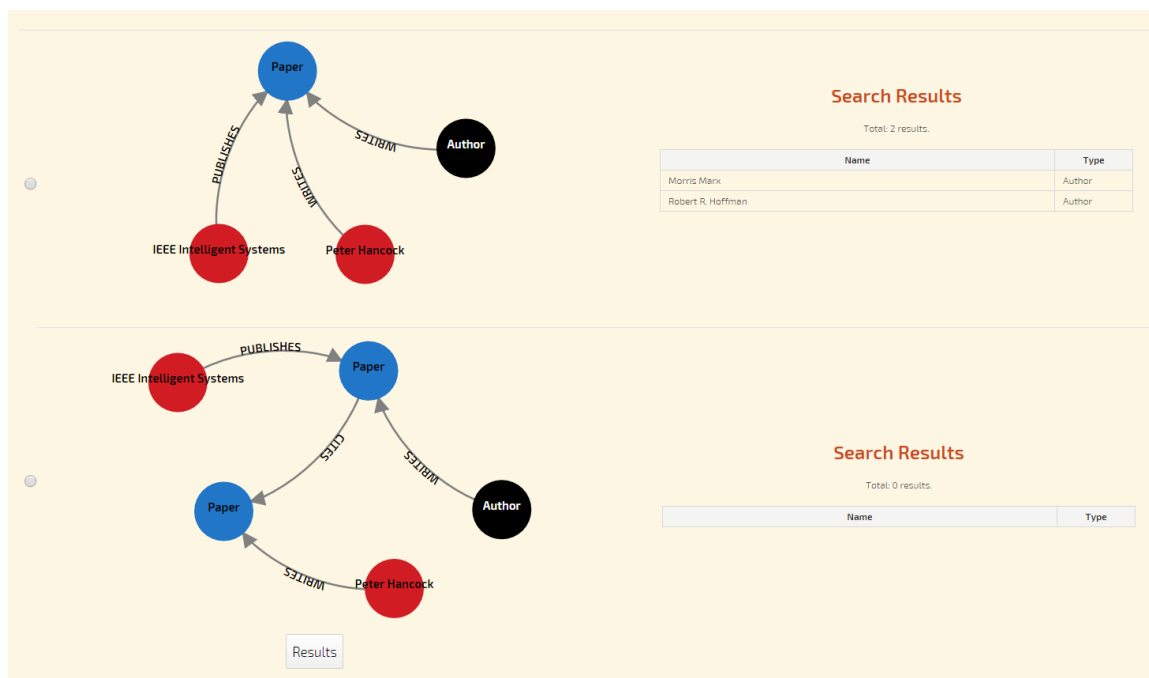


Figure 4.13: Two candidate graph queries and search of the query

4.3.2 Experiments

Due to the lack of readily available compiled resources, twenty test queries were handcrafted by the authors. The twenty test queries were constructed and divided into four groups based on the number of bibliographic nodes (i.e., from two to five) in a query. Because creating queries is an open problem, we were not able to apply standard measures such as inter-coder reliability, but performed several rounds of discussions to exclude subjectivity as much as possible. As a follow-up examination, we consulted three researchers to review the test queries. Researchers were asked to rank those queries that best match their everyday bibliographic information needs, and recommend additional queries that were not included in the original test queries. We selected top 10 queries ranked by the researchers (referred to as regular queries in Table 4.4). In addition, we

included five advanced queries of common bibliometric tasks such as bibliographic coupling, paper co-citation, author co-citation, co-author, and co-word. The final 15 test queries are shown in Appendix B. We benchmarked our system’s representability of these queries in relation to three major bibliographic information retrieval systems (i.e., Web of Science, Scopus, and Google Scholar). We measure the extent to which these queries are representable in one query and directly answerable without initiating other search tasks. For fair comparisons, we included the add-on features of the major systems (e.g., “Analyze Results” of Web of Science) and treated them as integrated components of the search tasks. Table 4.4 shows the ratio of queries that can be represented in each system with unanswerable queries listed in parentheses.

Table 4.4: The ratio of directly representable queries in each system

Test queries		Web of Science	Scopus	Google Scholar	Our system
Regular queries (unanswerable queries)		7/10 (No. 4, 5, & 10)	7/10 (No. 4, 5, & 10)	3/10 (No. 3, 4, 5, 7, 8, 9, and 10)	10/10 (None)
Advanced queries	Bibliographic coupling	N	N	N	Y
	Paper co-citation	N	N	N	Y
	Author co-citation	N	N	N	Y
	Co-author	Y	Y	Y	Y
	Co-word	Y	Y	N	Y

As shown in Table 4.4, existing systems have limitations in directly representing the information included in the test queries, despite the fact that we took their additional add-on features into consideration. Google Scholar did not fulfill majority of the tasks because it is not designed to retrieve entity types other than papers and does not explicitly manage metadata. Web of Science and Scopus performed better than Google Scholar, but were not able to represent a few test queries directly (i.e., bibliographic coupling, paper co-citation, and author co-citation). We were able to retrieve entity types other than papers by using their add-on features (e.g., “Analyze Results”) in Web of Science and Scopus, but the two systems do not support a feature where users can conduct a follow-up search through the add-on options. For example, items (e.g., authors) displayed in the “Analyze Results” feature are only used to refine the previous search results. Using the fourth test query as an example (“*Papers of authors who wrote Term-weighting Approaches in Automatic Text Retrieval*”), a general solution to this query involves getting author names of the paper and retrieving papers written by those papers. Web of Science supports getting author names by providing the title of the paper; however, it does not support the retrieval of papers written by those papers (though one can manually record the author names and initiate another round of search). For the advanced queries, the strength of our system is more evident. Our system was able to represent all the queries that are common for bibliometric tasks.

5. User-centered Evaluation

5.1 Experimental Setup

5.1.1 Overall Experimental Setup

The experimental setting is a laboratory study employing a mixed design (between-group and within subjects). Twenty participants were recruited using a convenience sampling method. A pilot-test was conducted to adjust the research setting. For example, through interviews with potential participants, we found that undergraduate students use academic information retrieval systems infrequently. This is probably due to their lack of research experience. Thus, the recruit is limited to people who have a Master's or a more advanced degree. Among the 20 participants, 10 are male and the other 10 are female. Nineteen participants are at the age of 25 to 34 while one participant falls in the range of 35 to 44. There are four Master's students, 13 doctoral students, two postdoctoral researchers, and one professor. Twelve participants are majored in Information Science, followed by Linguistics (three), Computer Science (three), and Biomedical Engineering (two).

5.1.2 Tasks and Measures

Tasks were designed as a known-item search, in which users were provided with partial information to perform a search. Known-item searches are a common type of bibliographic searches. For example, users search for papers with partial information

such as keywords and authors. Simulated work tasks (Borlund & Ingwersen, 1997) were employed to create the tasks in this study. As Borlund (2003) discussed, a simulated work task is a short cover story that describes situations where information needs arise. Because the three systems to be evaluated provide different ways of representing information needs, simulated situations are a good fit that helps us better understand how users formulate queries using different mechanisms. Overall, two task sets of simulated situations with varying task complexity (low and high) were created. The level of task complexity was decided based on the amount of effort needed to solve a problem. Participants were equally (i.e., ten for each set) divided into the two task sets. Each task set has three tasks (one for each system). To reduce the variance of using different tasks in each set, balanced task sets (e.g., Käksi & Aula, 2008) were created. Specifically, tasks in each task set were made as similar as possible by only modifying terms (e.g., author name) in the description. This helps reduce learning effects and variance of task complexity within each task set. A factorial design with six conditions (three system types by two levels of task complexity) was used. Therefore, each system was tested under the condition of both complexity levels of tasks and each complexity level of tasks was tested with all system types. Table 5.1 shows two example task situations (one in each level of task complexity). Questionnaires were used to elicit data from the participants. Both closed and open questions were asked in the questionnaires. Closed questions were largely based on a seven-point Likert scale. Five questionnaires: a background, three post-task, and an exit were used. A background questionnaire was used to elicit background information about the participants. Three post-task questionnaires

were administered after completing each of three tasks. The exit questionnaire was used to elicit overall cross-system comparisons and ratings.

Table 5.1: Two examples task situation used in the experiment

Example 1 (low-complexity task)	Example 2 (high-complexity task)
<ul style="list-style-type: none"> • John is a first-year doctoral student who is interested in Information Retrieval. His adviser asked him to find a few seminal papers and do a literature review. John knows Gerard Salton is one of the prominent researchers in the field. Among many journals, he plans to take a look at Communications of the ACM. • Please use the form-based system and find papers that were written by Gerard Salton and published in Communications of the ACM. 	<ul style="list-style-type: none"> • James is a first-year doctoral student who is interested in Data Mining. His advisor asked him to find a few seminal papers and do a literature review. James knows Jiawei Han is one of the prominent researchers in the field. After reviewing Jiawei Han's papers, James find that he is specifically interested in a subarea of Data Mining called clustering. James wants to find out more researchers who cited Jiawei Han's papers and do research on clustering. • Please use the natural language-based system and find out authors who wrote papers on clustering that cited papers that were written by Jiawei Han.

For evaluation measures, we adopted notions discussed in Kelly's study (2009). As discussed by Kelly, in IIR, unlike HCI research, performance is seen as a separate concept from usability and usability is usually evaluated based on self-report measures.

In this study, we evaluated the systems in terms of performance and usability. Performance was measured by a set of objective measures while usability was examined by analyzing data elicited from questionnaires. Performance was defined as system's ability of helping users resolve bibliographic information retrieval tasks. Table 5.2 shows measures of performance and their operational definitions.

Table 5.2: Performance measures used in the experiment

Category	Measure	Description
Relevance-based	Success rate	A binary measure (i.e., success or not) represents completion of each task.
Interaction-based	Search time	Time taken to complete a search task
	Query size	The number of issued queries to complete a search task.

Success rate is used to measure participants' performance on retrieving relevant search results. Participants' answers are matched against correct answers to see whether they are identical. Used synthetically, it shows the fraction of participants who complete the search tasks. A larger fraction shows that the participants perform better with one system than another. Search time measures the time a participant uses to complete a task. It largely depends on the time a user spends on formulating a query because the time used by the background processes (e.g., interpreting queries, querying the database) are almost identical to the extent that we can ignore the differences across the three systems. Shorter search time represents better performance. Query Size measures the number of queries a participant formulates to finish a task. Participants might formulate more than one query

to finish a task due to a lack of experience. A smaller query size denotes that the participants perform better in terms of formulating queries correctly.

ISO's definition of usability (ISO 9241-11, 1998) has three dimensions: effectiveness, efficiency, and satisfaction as shown in the following list (p. 2):

- *Effectiveness is the “accuracy and completeness with which users achieve specified goals.”*
- *Efficiency is the “resources expended in relation to the accuracy and completeness with which users achieve goals.”*
- *Satisfaction is the “freedom from discomfort, and positive attitudes of the user to the product.”*

Kelly (2009) discussed that measures for effectiveness and efficiency used in HCI (e.g., Hornbæk, 2006) typically overlap with the performance measures in IIR (e.g., precision and time taken to complete a task). Thus, in this paper, we test usability by measuring satisfaction with three additional measures including ease of use, ease of learning, and usefulness. We use the USE questionnaire (Lund, 2011) to measure the above four dimensions of usability, in which each item is measured using a seven-point Likert scale.

5.2 Results

5.2.1 Participants' Overall Experience

Through a background questionnaire, we collected information on participants' general experience on academic search systems. Among the 20 participants, eight

participants reported that they use academic search systems more than seven times in a week, followed by seven participants (one to two times), four participants (three to four times), and one participant (five to six times). About one third are active users of academic search systems; one third use academic search systems less frequently; and the remaining represent a moderate user group. As for the primarily used search system, 13 participants selected Google Scholar, five participants selected Web of Science, and two participants selected other systems (e.g., Microsoft Academic Search). Participants were also asked to rate their ability of using academic search systems through a five point Likert scale. Eleven participants responded by selecting average, followed by proficient (eight) and very proficient (one). In terms of the motivation of using academic search systems, six participants responded that they usually have specific information needs in mind before searching while three participants responded that their main purpose is exploring. Eleven participants responded with both choices. Given this, we argue that most participants use academic search systems with specific information in mind. In a question that asked current academic search systems' support of bibliographic information needs, 14 participants responded that they are not powerful enough in terms of representing complex bibliographic information need, and some of their bibliographic information needs are not representable in the current systems while six participants were satisfied with them.

5.2.2 Results of Performance Evaluation

Table 5.3 shows the overall results of the performance evaluation. We can see how two groups of participants performed for each of the three systems. As mentioned

previously, 10 were assigned to the low-complexity task and the other 10 were assigned to the high-complexity task. We show how many participants in each group successfully completed the tasks (success rate), and how long they spent on completing the tasks (search time, in seconds), and how many queries they formulated to represent information needs.

Table 5.3: Results on the performance evaluation

Task complexity	Measure	System type		
		Form-based	Natural language-based	Visual graph-based
Low	Success rate	10/10	10/10	10/10
	Search time	Max: 58, Min: 10 Mean: 26 , Median: 22	Max: 61, Min: 12 Mean: 33 , Median: 30	Max: 91, Min: 30 Mean: 61 , Median: 58
	Query Size	Max: 3, Min: 1 Mean: 1.5 , Median: 1	Max: 5, Min: 1 Mean: 2.3 , Median: 2	Max: 4, Min: 1 Mean: 1.7 , Median: 1
High	Success rate	9/10	10/10	7/10
	Search time	Max: 150, Min: 20 Mean: 70 , Median: 64	Max: 90, Min: 21 Mean: 53 , Median: 48	Max: 191, Min: 90 Mean: 135 , Median: 131
	Query size	Max: 3, Min: 1 Mean: 1.3 , Median: 1	Max: 5, Min: 1 Mean: 2 , Median: 1	Max: 5, Min: 1 Mean: 1.7 , Median: 1

For the low-complexity task set, all participants completed the tasks using all the systems while for the high-complexity task set, one participant failed the task for the form-based system and three participants failed for the visual graph-based system. A reasonable explanation is that the natural language-based system requires no additional

knowledge to use. On the other hand, users need to learn how to use the other two systems, which requires additional time and effort. Because the participants have never used a visual graph interface to formulate bibliographic queries, the visual graph-based system is less familiar to them. Thus, compared with the form-based system, which has been widely adopted by major academic search systems, the visual graph-based system has more features to be learned and the participants performed worse with the system than with the other systems.

The participants spent the least time (i.e., 26 secs) with the form-based system in the low-complexity task set while for the high-complexity task set, the least time (i.e., 53 secs) was achieved with the natural language-based system. We can see that, for simple tasks, the form-based system is the fastest way to formulate queries, probably due to its simplicity and users' familiarity with the system. As information needs become more complex, the natural language-based system proves to be a faster solution because one just needs to write down natural language queries without repeating mouse and keyboard operations such as selecting items and inputting values. In both task sets, the participants spent the most time with the visual graph-based system. This was expected because the visual graph-based system is specifically designed to answer complex bibliographic queries (e.g., bibliographic coupling and author co-citation), and it requires every single node in a query to be manually specified with properties.

Query size reveals the magnitude of errors made by the participants. In both task sets, the participants formulated the smallest number of queries (i.e., 1.5 and 1.3) with the form-based systems and the largest number of queries (2.3 and 2) with the natural language-based system. Typographical errors might be the factor that affected the

participants' query size in the natural language-based system. Before the evaluation, we expected that the largest query size would be recorded with the visual graph-based system because it is the most complex system among the three. However, in both task sets, the query size was between the other two systems. It is partly due to its query verification module that automatically corrects misrepresented queries and recommends new ones.

5.2.3 Results of Usability Evaluation

Figure 5.1 shows the bar charts of the four usability measures (i.e., usefulness, ease of use, ease of learning, and satisfaction). In the figure and tables following below, F denotes the form-based system, NL denotes the natural language-based system, and VG denotes the visual graph-based system. For each measure, the means and error bars of the two groups (high-complexity group vs. low-complexity group) are shown.

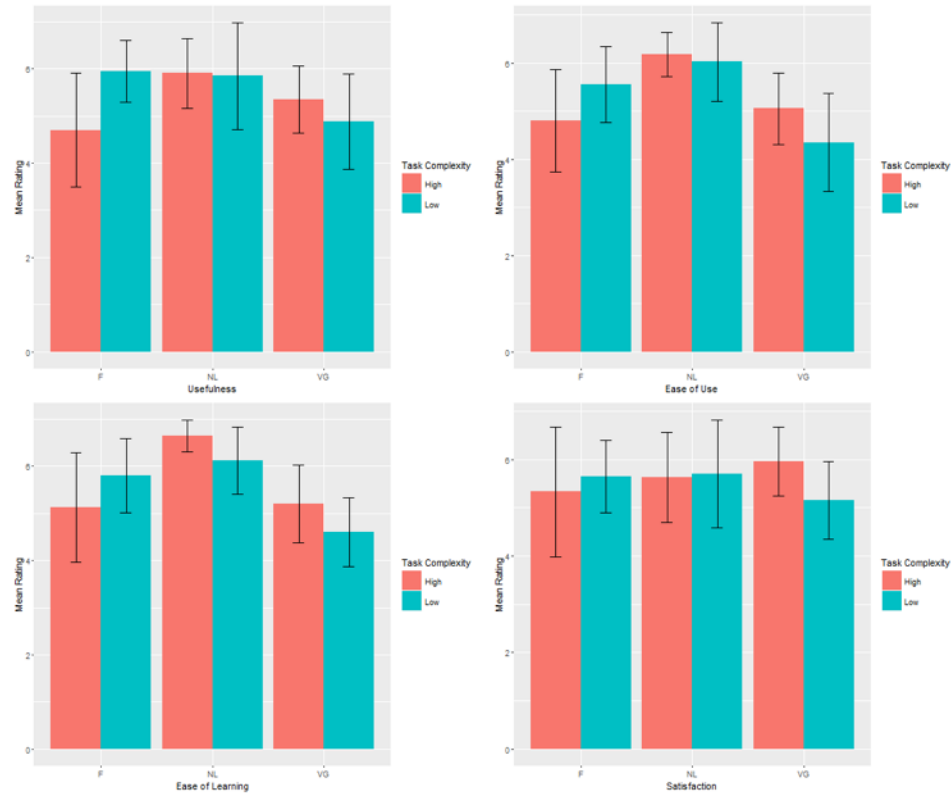


Figure 5.1: Descriptive statistics for usability measures

To further explore any statistically significant differences, we run a 2*3 mixed-design ANOVA because the task complexity is a between-group variable and system type is a repeated-measure variable. Table 5.4 shows the results.

Table 5.4: Results of the mixed-design ANOVA

Effect	Usefulness	Ease of Use	Ease of Learning	Satisfaction
Main effect of Task complexity	$F(1, 78) = 1.41$, $p = .24$, $\eta^2 = .007$	$F(1, 78) = .025$, $p = .87$, $\eta^2 < .001$	$F(1, 78) = .53$, $p = .47$, $\eta^2 = .004$	$F(1, 78) = .3$, $p = .58$, $\eta^2 = .002$
Main effect of System type	$F(2, 156) = 7.48$, $p < .001$, $\eta^2 = .05$	$F(2, 156) = 28.65$, $p < .001$, $\eta^2 = .16$	$F(2, 156) = 52.05$, $p < .001$, $\eta^2 = .21$	$F(2, 156) = .42$, $p = .65$, $\eta^2 = .002$
Interaction between the two	$F(2, 156) = 9.75$, $p < .001$, $\eta^2 = .07$	$F(2, 156) = 7.57$, $p < .001$, $\eta^2 = .05$	$F(2, 156) = 11.81$, $p < .001$, $\eta^2 = .06$	$F(2, 156) = 4.7$, $p = .01$, $\eta^2 = .03$

As shown in Table 5.4, there is no significant difference between ratings of the two groups of task complexity. However, we see participants rated the three systems differently when evaluating their usefulness, ease of use, and ease of learning (i.e., all of them are significant at the 0.001 level). Interactions between task complexity and system type are shown to be significant in all the four measures. Pairwise t-test with Bonferroni correction was performed to see how differently participants rated the systems (Table 5.5). However, because of the existence of the interaction effects between task complexity and system types, the ratings could also had been partly affected by task complexity. Satisfaction was not included because it is not significant in Table 5.4.

Table 5.5: Results of pairwise t-test on system type

Usefulness			Ease of Use			Ease of Learning		
	F	NL		F	NL		F	NL
NL	0.0168*	-	NL	<0.0001***	-	NL	<0.0001***	-
VG	1.00	0.0021**	VG	0.12	<0.0001***	VG	0.0069**	<0.0001***

Based on Table 5.5 and Figure 5.1, we see that the participants rated the natural language-based system more useful than the other two systems. It was also rated as the system that is the most easy to use and learn. The participants also reported that the form-based system was easier to learn than the visual graph-based system. In terms of usefulness and ease to use, we do not see any significant difference between the form- and visual graph-based systems. Finally, simple effects analysis was performed to understand the interaction between task complexity and system type. The datasets of four measures were further split into three subsets. Then, ANOVA and Tukey's tests were

performed to see whether significant differences exist between the two complexity groups upon each of the three system types. The results are shown in Table 5.6.

Table 5.6: Results of ANOVA and Tukey's tests

System type	Usefulness	Ease of Use	Ease of Learning	Satisfaction
F (Low vs. High)	0.3594	0.0227*	0.0341*	0.3594
NL (Low vs. High)	0.8267	0.5567	0.0058**	0.8267
VG (Low vs. High)	0.0068**	0.0345*	0.038*	0.0068**

Combining the result shown in Table 5.6 with that of Figure 5.1, we see that participants assigned to the high-complexity tasks rated the visual graph-based system higher than the participant group of low-complexity task in usefulness and satisfaction. This shows that participants thought the visual graph-based system is more useful and satisfactory for high-complexity tasks. For the form-based system, the participant group of low-complexity tasks rated it higher than its counterparts in terms of ease of use and learning. The natural language-based system was reported with similar ratings on ease of use by the two groups. This is the strength of the natural language-based system that no additional effort is needed to formulate more complex queries, and there is no significant difference of effort between formulating low- and high-complexity queries. A seemingly contradictory result is that participants assigned to the high-complexity tasks rated both the natural language- and visual graph-based systems higher than the participant group of low-complexity task in ease of learning. Because participants in the high-complexity task group interacted with all the three systems, who felt the form-based system hard to learn (e.g., due to its unsuitability to high-complexity task) might feel the other two systems

relatively easier to learn. Overall, the interaction effect is the most significant in the visual graph-based system and least significant in the natural language-based system.

6. Conclusion and Future Work

6.1 Conclusion

In the dissertation, we proposed, implemented, and evaluated three graph-based interactive bibliographic information retrieval systems. Compared with current well-known bibliographic information retrieval systems, our system has several advantages: (a) they support searching for various types of bibliographic information and deliver these types as end search results; (b) they support easy ways of representing complex bibliographic queries, which is not readily available in existing bibliographic information retrieval systems; (c) they provide interactive user interfaces for users to refine queries; and (d) they expedite the query time by adopting a graph data model.

The study also recruited 20 participants to compare three systems from a variety of aspects such as success rate, search time, query size, usefulness, ease of use, ease of learning, and satisfaction. We employed a mixed-design experiment to understand how task complexity and system type affect users' performance and usability ratings.

The form-based system needed the least time (i.e., 26 secs for the low-complexity tasks) to formulate simple bibliographic queries and allowed the participants to make the least mistakes (i.e., the average query size 1.5 and 1.3 for low- and high-complexity tasks). Participants stated that "The form-based system makes everything clear, so it is easy for me to handle each part of my query", "The form-based system is kind of fun and may come into handy pretty fast". One weakness of the system is that, with complex queries, participants feel that it is not as easy to use as it is with simple queries. The

participant group of low-complexity task rated higher than its counterparts in terms of ease of use and learning. A participant assigned to high-complexity tasks stated that “The form-based (system) needs users to take more logic consideration and have more clicking.”

The natural language-based system needed the least time (i.e., 53 secs) to formulate high-complexity bibliographic queries and was rated as the most useful, easy-to-use, and easy-to-learn system. Participants stated that “Using natural language for searching bibliographic information would be the most intuitive way”, “As long as my query is in good grammar, the system does everything automatically and transforms it into a graphical query, which is quite cool, and time-efficient”. Flexibility was reported as the main weakness of the system. A participant stated that “The natural language-based system is easy to use, but it didn't allow too much flexibility on how to express a query by natural language”. This was expected because the system requires users to input queries without grammatical errors and state-of-the-art NLP parsers can sometimes make mistakes even though the input queries are correct.

The visual graph-based system did not get the best ratings in any of the measures. However, its strengths lie in the support of complex queries as shown that it got better ratings of usefulness and satisfaction by the high-complexity task group. Participants stated that “The visual graph-based system is a very flexible one. It seems everything is within my control. I felt comfortable to use this tool”, “The visual system seems a little bit more complex than the other two systems, but it is actually the most powerful one that can answer lots of, almost every query that I can imagine.” As stated by a participant- “The visual graph-based system has a learning curve and requires some learning costs in

the beginning”, one weakness of the system is that the system requires more effort than the other two systems to get familiar with it. The participants rated the system as the most hard-to-learn system.

Based on the evaluation results, we conclude that each system has their own strengths and weaknesses. In the background questionnaire, 14 out of 20 participants stated that current bibliographic search systems are not powerful enough in terms of representing and answering complex bibliographic information need. It is clear that there is no single answer to address this limitation, and different approaches have different values as shown in the evaluation results. Our contribution lies in the design, implementation, and evaluation of these kinds of systems. We believe the proposed systems are effective and efficient solutions for addressing complex bibliographic information needs. In addition, we believe the experimental design and results shown in this paper can serve as a useful guideline and benchmark for future studies.

6.2 Future Work

Designing bibliographic information retrieval systems for complex bibliographic queries is a new research field. There are many unexplored territories and challenging research issues. Here we illustrate a few of them.

1) Unified Evaluation Frameworks for Complex Bibliographic Information Retrieval

Complex bibliographic information retrieval has not been widely studied. Even though we contributed test queries and a few benchmarks to the field, there is a lack of unified evaluation frameworks that facilitates evaluation and comparison of peer systems.

Researchers can build evaluation frameworks by collecting a larger number of test queries using crowdsourcing and providing unified application interface.

2) Personalized Bibliographic Information Retrieval

In general, a user is only interested in a small portion of the entire bibliographic data. For example, a user who does research on social science rarely looks for papers about quantum physics. Even in one domain, a user is mostly interested in a few research topics. It is both efficient and effective to provide personalized bibliographic information retrieval in which the system learns about users based on their previous search history. Personalized bibliographic information retrieval is time-efficient because we can detect and search through only a subset of the original dataset that might be the interest of a specific user. It is also effective in many cases. For example, assume there are multiple people with the name- “John” and a user entered a query “papers by John”, then personalized bibliographic information retrieval can detect which John the user is searching for based on the user’s previous history such as domain and research interests. Therefore, analyzing previous queries generated by a user and providing personalized search results is an important research issue.

3) Dynamic Clustering of Search Results

Search results of complex bibliographic information retrieval are more specific than that of traditional bibliographic information retrieval. Traditional information retrieval systems provide predefined categories to classify and refine search results. However, this approach is not very efficient in complex bibliographic information retrieval because most categories are meaningless due to relatively small number of search results and the specificity of search results. Dynamic clustering denotes cluster search results by

dynamically applying cluster criteria. For example, the search results of the query “*papers on information retrieval*” can be clustered based on collaboration among researchers and produce results of so-called “school of thought”. However, the same criteria cannot be effectively applied to the case of “*papers by John*”, because all the papers in the search results are written by the same author. Here, we need to apply different criteria. Therefore, dynamic clustering of search results is an ideal approach for complex bibliographic information retrieval.

List of References

- Abacha, A. B., & Zweigenbaum, P. (2015). MEANS: A medical question-answering system combining NLP techniques and semantic web technologies. *Information Processing & Management*, 51(5), 570-594. doi:10.1016/j.ipm.2015.04.006
- Aggarwal, C. C., & Wang, H. (Ed.). (2010). *Managing and mining graph data* (Vol. 40). New York: Springer.
- Aghaei Chadegani, A., Salehi, H., Yunus, M. M., Farhadi, H., Fooladi, M., Farhadi, M., & Ale Ebrahim, N. (2013). A comparison between two main academic literature collections: Web of Science and Scopus databases. *Asian Social Science*, 9(5), 18–26.
- Al-Maskari, A., & Sanderson, M. (2010). A review of factors influencing user satisfaction in information retrieval. *Journal of the American Society for Information Science and Technology*, 61(5), 859-868. doi:10.1002/asi.21300
- Al-Maskari, A., & Sanderson, M. (2011). The effect of user characteristics on search effectiveness in information retrieval. *Information Processing and Management*, 47(5), 719-729. doi:10.1016/j.ipm.2011.03.002
- Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases—an introduction. *Natural language engineering*, 1(01), 29-81.
- Angles, R., & Gutierrez, C. (2008). Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1), 1.
- Baeza-Yates, R., Brisaboa, N., & Larriba-Pey, J. (2010). A model for automatic generation of multi-partite graphs from arbitrary data. (pp. 49-60). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-642-16720-1_5
- Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American*, 284(5), 28-37.
- Bikel, D. M., Miller, S., Schwartz, R., & Weischedel, R. (1997). Nymble: a high-performance learning name-finder. In *Proceedings of the fifth conference on applied natural language processing* (pp. 194-201). Association for Computational Linguistics.
- Borlund, P. (2003). The IIR evaluation model: A framework for evaluation of interactive information retrieval systems. *Information Research-an International Electronic Journal*, 8(3), 152.
- Borlund, P. (2016). A study of the use of simulated work task situations in interactive information retrieval evaluations: A meta-evaluation. *Journal of Documentation*, 72(3), 394-413.
- Borlund, P., & Ingwersen, P. (1997). The development of a method for the evaluation of interactive information retrieval systems. *Journal of Documentation*, 53(3), 225-250. doi:10.1108/EUM0000000007198

- Borlund, P., & Schneider, J. (2010, August). Reconsideration of the simulated work task situation: A context instrument for evaluation of information retrieval interaction. Paper presented at the Third Information Interaction in Context Symposium (IiX), 155-164. doi:10.1145/1840784.1840808
- Broekstra, J., Kampman, A., & Van Harmelen, F. (2002). Sesame: A generic architecture for storing and querying RDF and RDF schema. In *The Semantic Web—ISWC 2002* (pp. 54–68). Berlin Heidelberg:Springer, doi:10.1007/3-540-48005-6_7.
- Cafarella, M. J., & Etzioni, O. (2005, May). A search engine for natural language applications. In *Proceedings of the 14th international conference on World Wide Web* (pp. 442-452). ACM.
- Carroll, J. J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., & Wilkinson, K. (2004). Jena: Implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters* (pp. 74–83). New York: ACM.
- Ceri, S., Comai, S., Damiani, E., & Fraternali, P. (1999). XML-GL: A graphical language for querying and restructuring XML documents. *Computer Networks*, 31(11–16), 1171.
- Chadegani, A. A., Salehi, H., Yunus, M. M., Farhadi, H., Fooladi, M., Farhadi, M., & Ebrahim, N. A. (2013). A comparison between two main academic literature collections: Web of science and Scopus databases. *Asian Social Science*, 9(5), 18-26.
- Chieu, H. L., & Ng, H. T. (2002). Named entity recognition: a maximum entropy approach using global information. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1* (pp. 1-7). Association for Computational Linguistics.
- Cimiano, P., Haase, P., Heizmann, J., Mantel, M., & Studer, R. (2008). Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system. *Data and Knowledge Engineering*, 65(2), 325–354.
- Clemmensen, M. L., & Borlund, P. (2016). Order effect in interactive information retrieval evaluation: An empirical study. *Journal of Documentation*, 72(2), 194-213. doi:10.1108/JD-04-2015-0051
- Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387.
- Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics* (pp. 16-23). Association for Computational Linguistics.
- Cook, D. J., & Holder, L. B. (Eds.). (2006). *Mining graph data*. John Wiley & Sons, Hoboken, New Jersey.
- Damljanovic, D., Agatonovic, M., & Cunningham, H. (2010). Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *The semantic web: Research and applications* (pp. 106-120). Springer Berlin Heidelberg.

- De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC* (Vol. 6, No. 2006, pp. 449-454).
- de Ribaupierre, H. (2014). *Precise information retrieval in semantic scientific digital libraries* (Doctoral dissertation). University of Geneva, Geneva, Switzerland.
- Derczynski, L., Maynard, D., Rizzo, G., Erp, M. v., Gorrell, G., Troncy, R., Bontcheva, K. (2015). Analysis of named entity recognition and linking for tweets. *Information Processing & Management*, 51(2), 32-49. doi:10.1016/j.ipm.2014.10.006
- Dominguez-Sal, D., Urbón-Bayes, P., Giménez-Vanó, A., Gómez-Villamor, S., Martínez-Bazan, N., & Larriba-Pey, J. L. (2010, July). Survey of graph database performance on the hpc scalable graph analysis benchmark. In *International Conference on Web-Age Information Management* (pp. 37-48). Springer Berlin Heidelberg.
- Earley, J. (1970). An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2), 94-102.
- Erwig, M. (2003). Xing: A visual XML query language. *Journal of Visual Languages and Computing*, 14(1), 5–45. [http://dx.doi.org/10.1016/S1045-926X\(02\)00074-5](http://dx.doi.org/10.1016/S1045-926X(02)00074-5)
- Falagas, M. E., Pitsouni, E. I., Malietzis, G. A., & Pappas, G. (2008). Comparison of PubMed, Scopus, web of science, and Google scholar: Strengths and weaknesses. *The FASEB Journal*, 22(2), 338–342.
- Fazzinga, B., & Lukasiewicz, T. (2010). Semantic search on the web. *Semantic Web*, 1(1–2), 89–96.
- Fersini, E., Messina, E., Felici, G., & Roth, D. (2014). Soft-constrained inference for named entity recognition. *Information Processing & Management*, 50(5), 807-819. doi:10.1016/j.ipm.2014.04.005
- Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 363-370). Association for Computational Linguistics.
- Geuna, A., & Martin, B. R. (2003). University research evaluation and funding: An international comparison. *Minerva*, 41(4), 277–304.
- Gómez-Villamor, S., Soldevila-Miranda, G., Giménez-Vañó, A., Martínez-Bazan, N., Muntés-Mulero, V., & Larriba-Pey, J. (2008). BIBEX: A bibliographic exploration tool based on the DEX graph query engine. *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, 735–739. <http://dx.doi.org/10.1145/1353343.1353439>
- Guha, R., McCool, R., & Miller, E. (2003). Semantic search. In *Proceedings of the 12th International Conference on World Wide Web* (pp. 700–709). ACM, New York, NY.
- Gusfield, D. (1997). *Algorithms on strings, trees, and sequences: Computer science and computational biology*. New York; Cambridge [England];: Cambridge University Press.

- Habernal, I., & Konopík, M. (2013). SWSNL: Semantic web search using natural language. *Expert Systems with Applications*, 40(9), 3649-3664.
- Hall, M. M., & Toms, E. (2013, September). Building a Common Framework for IIR Evaluation. Paper presented at the Forth International Conference of the Cross-Language Evaluation Forum for European Language, 17-28. doi: 10.1007/978-3-642-40802-1_3
- He, H., & Singh, A. K. (2008). Graphs-at-a-time: Query language and access methods for graph databases. In *Proceedings of the 2008 ACM SIGMOD International conference on Management of data* (pp. 405–418).
- Hogenboom, F., Milea, V., Frasincar, F., & Kaymak, U. (2010). RDF-GL: A SPARQL-based graphical query language for RDF. In *Emergent web intelligence: advanced information retrieval*. pp. 87–116. London: Springer
- Holzschuher, F., & Peinl, R. (2013). Performance of graph query languages: Comparison of cypher, gremlin and native access in Neo4j. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops* (pp. 195–204). ACM, New York, NY.
- Hornbæk, K. (2006). Current practice in measuring usability: Challenges to usability studies and research. *International Journal of Human - Computer Studies*, 64(2), 79-102. doi:10.1016/j.ijhcs.2005.06.002
- Huan, J., Wang, W., Prins, J., & Yang, J. (2004). Spin: Mining maximal frequent subgraphs from graph databases. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 581–586). ACM, New York, NY.
- ISO 9241-11 (1998). Ergonomic Requirements of Office Work with Visual Display Terminals (VDTs): Part II: Guidance on usability. International Organization for Standardization
- Jacso, P. (2005). As we may search-Comparison of major features of the Web of Science, Scopus, and Google Scholar citation-based and citation-enhanced databases. *CURRENT SCIENCE-BANGALORE-*, 89(9), 1537.
- Jiang, H., Wang, H., Yu, P. S., & Zhou, S. (2007). Gstring: A novel approach for efficient search in graph databases. In *IEEE 23rd international conference on data engineering, 2007. ICDE 2007* (pp. 566–575).
- Kacholia, V., Pandit, S., Chakrabarti, S., Sudarshan, S., Desai, R., & Karambelkar, H. (2005). Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on very large data bases* (pp. 505–516).
- Käki, M., & Aula, A. (2008). Controlling the complexity in comparing search user interfaces via user studies. *Information Processing and Management*, 44(1), 82-91. doi:10.1016/j.ipm.2007.02.006
- Kaufmann, E., & Bernstein, A. (2010). Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4), 377-393.

- Kaufmann, E., Bernstein, A., & Fischer, L. (2007). NLP-reduce: A “naïve” but domain independent natural language interface for querying ontologies. In 4th European Semantic Web Conference (ESWC 2007) (pp. 1–2).
- Kelly, D. (2009). Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval*, 3(1–2), 1-224. doi:10.1145/2808194.2809465
- Kelly, D., Arguello, J., Edwards, A., & Wu, W. C. (2015, September). Development and evaluation of search tasks for IIR experiments using a cognitive complexity framework. Paper presented at the ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR), 101-110
- Kelly, D., & Sugimoto, C. R. (2013). A systematic review of interactive information retrieval evaluation studies, 1967–2006. *Journal of the American Society for Information Science and Technology*, 64(4), 745-770. doi:10.1002/asi.22799
- Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1* (pp. 423-430). Association for Computational Linguistics.
- Klein, D., & Manning, C. D. (2004). Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics* (p. 478). Association for Computational Linguistics.
- Lahdenmaki, T., & Leach, M. (2005). *Relational database index design and the optimizers*. John Wiley & Sons, Hoboken, New Jersey.
- Lei, Y., Uren, V., & Motta, E. (2006). Semsearch: A search engine for the semantic web. In *Managing knowledge in a world of networks* (pp. 238–245). Berlin Heidelberg: Springer, doi:10.1007/11891451_22.
- Li, F., & Jagadish, H. V. (2014). Constructing an interactive natural language interface for relational databases. *Proceedings of the VLDB Endowment*, 8(1), 73-84.
- Li, Y. (2009). Exploring the relationships between work task and search task in information search. *Journal of the American Society for Information Science and Technology*, 60(2), 275-291. doi:10.1002/asi.20977
- Li, Y., & Belkin, N. J. (2008). A faceted approach to conceptualizing tasks in information seeking. *Information Processing and Management*, 44(6), 1822-1837. doi:10.1016/j.ipm.2008.07.005
- Li, Y., & Belkin, N. J. (2010). An exploration of the relationships between work task and interactive information search behavior. *Journal of the American Society for Information Science and Technology*, 61(9), 1771-1789. doi:10.1002/asi.21359
- Li, Y., Wang, Y., & Huang, X. (2007). A relation-based search engine in semantic web. *IEEE Transactions on Knowledge and Data Engineering*, 19(2), 273–282.
- Liu, J., Cole, M., Liu, C., Bierig, R., Gwizdka, J., Belkin, N., Zhang, J., & Zhang, X. (2010, June). Search behaviors in different task types. Paper presented at the Joint Conference on Digital Libraries (JCDL), 69-78. doi:10.1145/1816123.1816134

- Lund, A. M. (2011). Measuring usability with the USE questionnaire. *Usability Interface*, 8(2), 3-6.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval* (Vol. 1, p. 496). Cambridge: Cambridge university press.
- Marcus, M. P., Marcinkiewicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2), 313-330.
- McCallum, A., & Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4* (pp. 188-191). Association for Computational Linguistics.
- Miller, E. (1998). An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1), 15–19.
- Mishra, P., & Eich, M. H. (1992). Join processing in relational databases. *ACM Computing Surveys (CSUR)*, 24(1), 63–113.
- Mu, X., Lu, K., & Ryu, H. (2014). Explicitly integrating MeSH thesaurus help into health information retrieval systems: An empirical user study. *Information Processing & Management*, 50(1), 24-40. doi:10.1016/j.ipm.2013.03.005
- Nadeau, D., & Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1), 3-3. doi:10.1075/li.30.1.03nad
- Ni, W., & Ling, T. W. (2003). GLASS: A graphical query language for semi-structured data. In *Eighth international conference on database systems for advanced applications. (DASFAA 2003)* (pp. 363–370).
- Park, C., & Lim, S. (2015). Efficient processing of keyword queries over graph databases for finding effective answers. *Information Processing & Management*, 51(1), 42.
- Rajbhandari, P., Shah, R. C., & Agarwal, S. (2012). Graph database model for querying, searching and updating. In *International Conference on Software and Computer Applications (ICSCA)*.
- Rau, L. F. (1991). Extracting company names from text. In *Artificial Intelligence Applications, 1991. Proceedings. Seventh IEEE Conference on* (Vol. 1, pp. 29-32). IEEE.
- Renaud, G., & Azzopardi, L. (2012, August). SCAMP: A tool for conducting interactive information retrieval experiments. Paper presented at the *Fourth Information Interaction in Context Symposium (IiX)*, 286-289. doi:10.1145/2362724.2362776
- Rindflesch, T. C., Tanabe, L., Weinstein, J. N., & Hunter, L. (2000). EDGAR: extraction of drugs, genes and relations from the biomedical literature. In *Pac Symp Biocomput* (Vol. 5, pp. 514-25).
- Robinson, I., Webber, J., & Eifrem, E. (2013). *Graph databases*. O'Reilly Media, Inc, Sebastopol, CA.
- Rohloff, K., Dean, M., Emmons, I., Ryder, D., & Sumner, J. (2007). An evaluation of triple-store technologies for large data stores. In *On the move to meaningful internet*

- systems 2007: OTM 2007 Workshops (pp. 1105–1114). Berlin Heidelberg: Springer, doi:10.1007/978-3-540-76890-6_38.
- Roy, S., & Zeng, W. (2013). Cognitive canonicalization of natural language queries using semantic strata. *ACM Transactions on Speech and Language Processing (TSLP)*, 10(4), 1-30. doi:10.1145/2539053
- Ryu, P., Jang, M., & Kim, H. (2014). Open domain question answering using wikipedia-based knowledge model. *Information Processing & Management*, 50(5), 683-692. doi:10.1016/j.ipm.2014.04.007
- Sakr, S., & Pardede, E. (2012). *Graph data management: Techniques and applications*. Hershey, PA: Information Science Reference.
- Salton, G. (1970). Evaluation problems in interactive information retrieval. *Information Storage and Retrieval*, 6(1), 29-44. doi:10.1016/0020-0271(70)90011-2
- Schweiger, D., Trajanoski, Z., & Pabinger, S. (2014). SPARQLGraph: A web-based platform for graphically querying biological semantic web databases. *BMC Bioinformatics*, 15(1), 279. <http://dx.doi.org/10.1186/1471-2105-15-279>Sun
- Score, S. C. (2009). Web of Science and Scopus: A comparative review of content and searching capabilities. *The Charleston Advisor*, 11, 5–18.
- Singhal, A. (2012). Introducing the knowledge graph: Things, not strings. Official Google Blog, Retrieved at <https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html>.
- Smucker, M. D., & Jethani, C. P. (2010, August). Impact of retrieval precision on perceived difficulty and other user measures. Paper presented at the Fourth Symposium on Human-Computer Interaction and Information Retrieval (HCIR), 20-23.
- Sun, Y., Barber, R., Gupta, M., Aggarwal, C. C., & Han, J. (2011). Co-author relationship prediction in heterogeneous bibliographic networks. In 2011 international conference on advances in social networks analysis and mining (ASONAM) (pp. 121–128).
- Sun, Y., Yu, Y., & Han, J. (2009). Ranking-based clustering of heterogeneous information networks with star network schema. Paper presented at the 797-806. doi:10.1145/1557019.1557107
- Tablan, V., Damljanovic, D., & Bontcheva, K. (2008). A natural language query interface to structured information. (pp. 361-375). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-540-68234-9_28
- Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., & Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 990–998). ACM, New York, NY.
- Tran, T., Cimiano, P., Rudolph, S., & Studer, R. (2007). Ontology-based interpretation of keywords for semantic search (pp. 523–536). Berlin Heidelberg: Springer.

- Tsuruoka, Y., & Tsujii, J. I. (2003). Boosting precision and recall of dictionary-based protein name recognition. In *Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine-Volume 13* (pp. 41-48). Association for Computational Linguistics.
- Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y., & Wilkins, D. (2010). A comparison of a graph database and a relational database: a data provenance perspective. In *Proceedings of the 48th annual Southeast regional conference* (p. 42). ACM.
- Wang, C., Xiong, M., Zhou, Q., & Yu, Y. (2007). Panto: A portable natural language interface to ontologies. In *The Semantic Web: Research and Applications* (pp. 473-487). Springer Berlin Heidelberg.
- Wei, X., Zhang, Y., & Gwizdka, J. (2014, August). YASFIIRE: Yet another system for IIR evaluation. Paper presented at the Fifth Information Interaction in Context Symposium (IiX). 316-319. doi:10.1145/2637002.2637051
- Wildemuth, B., & Freund, L. (2012, October). Assigning search tasks designed to elicit exploratory search behaviors. Paper presented at the Sixth Symposium on Human-Computer Interaction and Information Retrieval (HCIR), 1-10. doi:10.1145/2391224.2391228
- Wildemuth, B., Freund, L., & Toms, E. (2014). Untangling search task complexity and difficulty in the context of interactive information retrieval studies. *Journal of Documentation*, 70(6), 1118-1140. doi:10.1108/JD-03-2014-0056
- Williams, D. W., Huan, J., & Wang, W. (2007). Graph database indexing using structured graph decomposition. In *Data engineering, 2007. ICDE 2007. IEEE 23rd International Conference on* (pp. 976-985). IEEE, Piscataway, New Jersey.
- Yan, E., & Ding, Y. (2009). Applying centrality measures to impact analysis: A coauthorship network analysis. *Journal of the American Society for Information Science and Technology*, 60(10), 2107-2118.
- Yan, X., Yu, P. S., & Han, J. (2005). Substructure similarity search in graph databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 766-777). ACM, New York, NY.
- Ykhlef, M., & Alqahtani, S. (2009). GQLX: A new graphical query language for XML data. *Proceedings of the 11th international conference on information integration and web-based applications & services*, 201-208.
- Yuan, Y., Wang, G., Chen, L., & Wang, H. (2013). Efficient keyword search on uncertain graph data. *IEEE Transactions on Knowledge and Data Engineering*, 25(12), 2767-2779.
- Zhou, Q., Wang, C., Xiong, M., Wang, H., & Yu, Y. (2007). SPARK: Adapting keyword query to semantic search (pp. 694-707). Berlin Heidelberg: Springer.
- Zou, L., Chen, L., & Özsu, M. T. (2009). Distance-join: Pattern match query in a large graph database. *Proceedings of the VLDB Endowment*, 2(1), 886-897.

Zucon, G., Leelanupab, T., Whiting, S., Yilmaz, E., Jose, J. M., & Azzopardi, L. (2013;2012;). Crowdsourcing interactions: Using crowdsourcing for evaluating interactive information retrieval systems. *Information Retrieval*, 16(2), 267-305. doi:10.1007/s10791-012-9206-z

Appendix A: Natural Language Queries Tested in the Experiment

1. Papers by Gerard Salton
2. Michael Lawrence's papers
3. Papers that were written by Sangjun Lee
4. Papers about ontology
5. Authors of Automatic text structuring experiments
6. Papers that were cited by Energy-Aware and Time-Critical Geo-Routing in Wireless Sensor Networks
7. Terms of Opacity generalised to transition systems
8. Organization of Johann Eder
9. Sources that published The Effect of Faults on Network Expansions
10. Papers that were published in Theoretical Computer Science
11. Papers about classification and DNA
12. Papers that were written by John R. Mick and published in ACM SIGMICRO Newsletter
13. Papers cites papers that were written by Braham Barkat
14. Papers about modulation which were published in Neural Networks
15. Authors of University713 who wrote A control word model for detecting conflicts between microoperations
16. Sources that published Zesheng Chen's papers
17. Authors whose papers were published in AI Communications
18. Authors who wrote papers that were about simulation

19. Terms of Junghyun Nam's papers
20. Organizations of authors of A New Quadtree Decomposition Reconstruction Methods
21. Papers about survey, semantic, and retrieval
22. Authors of papers that were cited by papers that were published in Decision Support Systems
23. Papers that cite papers that were written by Rainer Engelke and published in Microsystem Technologies
24. Nina Yevtushenko's papers that were cited by papers that were written by Sergey Buffalov
25. Sources that published papers about genome and mining
26. Terms of Rafae Bhatti's papers that were published in Communications of the ACM
27. Sources that published Tomasz Jurdzinski's papers which are about automata
28. Terms of papers that were written by authors at University123
29. Organizations of authors whose papers were published in Journal of Multivariate Analysis
30. Authors who are affiliated with University007 and wrote papers about clustering
31. Papers about classification, which were cited by Asoke K. Nandi 's papers that had been presented in Pattern Recognition
32. Authors of papers that were cited by papers that were written by Changqiu Jin and published in Journal of Computational Physics
33. Terms of papers that were cited by papers about kernel and regression
34. Sources that published papers cited papers about middleware and embedded

35. Organizations of authors whose papers were cited by papers that were published in Journal of Robotic Systems
36. Organizations of authors who wrote papers on person similarity and bayesian
37. Papers about bayesian and electron which were written by authors at University170
38. Sources of papers, which were about eigenvalue and written by authors at University40
39. Authors at University899, who wrote papers that were about classifier, which were published in Applied Intelligence
40. Terms of papers that were published in Cybernetics and Systems Analysis and written by authors at University362

Appendix B: Fifteen queries tested in the experiment

1. Regular queries
2. Papers written by Gerard Salton
3. Papers on the topic of Human-Computer Interaction
4. Papers that were cited by “Introduction to Modern Information Retrieval”
5. Papers of authors who wrote “Term-weighting Approaches in Automatic Text Retrieval”
6. Papers on the topics of “The Hadoop distributed file system”
7. Papers that were written by Christopher D Manning and published by Association for Computational Linguistics
8. Authors at CMU, who presented papers in SIGCHI
9. Authors who wrote papers on the topic of information seeking behavior, which were presented in SIGIR
10. Topics of papers that were presented in SIGKDD
11. Conferences that presented papers on the topics discussed in “MapReduce: simplified data processing on large clusters”

Advanced queries

1. Bibliographic coupling (papers that were cited by “MapReduce: simplified data processing on large clusters” and “TheHadoop distributed file system”)
2. Paper co-citation (papers that cited both “MapReduce: simplified data processing on large clusters” and “The Hadoopdistributed file system”)

3. Author co-citation (papers that cited both Gerard Salton and James Allan)
4. Co-author (authors who co-authored with Gerard Salton)
5. Co-word (keywords that co-occurred with big data)

Vita

Yongjun Zhu

Education

- Ph.D. in Information Studies, Drexel University, 2013-2017
- M.S. in Industrial Engineering, Yonsei University, 2010-2012
- B.S. in Economics, Yanbian University of Science and Technology, 2005-2009

Research Areas

Information Retrieval, Data Mining, Science of Science, Health Informatics

Publications

- Yan, E. & Zhu, Y. (2017). Adding the dimension of knowledge trading to source impact assessment: Approaches, indicators, and implications. *Journal of the Association for Information Science & Technology*.
- Zhu, Y., Kim, M.C., & Chen, C. (2017). An investigation of the intellectual structure of opinion mining research. *Information Research*.
- Zhu, Y. & Yan, E. (2016). Searching bibliographic data using graphs: A visual graph query interface. *Journal of Informetrics*, 10(4), 1092-1107.
- Choi, N., Song, I.-Y., & Zhu, Y. (2016). A Model-based Method for Information Alignment: A Case Study on Educational Standards. *Journal of Computing Science and Engineering*, 10(3), 85-94.
- Zhu, Y., Yan, E., & Song, M. (2016). Understanding the evolving academic landscape of library and information science through faculty hiring data. *Scientometrics*, 108(3), 1461-1478.
- Zhu, Y., Song, M., & Yan, E. (2016). Identifying Liver Cancer and Its Relations with Diseases, Drugs, and Genes: A Literature-based Approach. *PLoS ONE*, 11(5), e0156091.
- Zhu, Y., Yan, E., & Song, I.-Y. (2016). The use of a graph-based system to improve bibliographic information retrieval: System design, implementation, and evaluation. *Journal of the Association for Information Science & Technology*
- Kim, M.C., Zhu, Y., & Chen, C. (2016). How are they different? A quantitative domain comparison of information visualization and data visualization (2000-2014). *Scientometrics*, 107(1), 123-165.
- Song, I.-Y. & Zhu, Y. (2015). Big data and data science: what should we teach? *Expert Systems*, 33(4), 364-373.

- Yan, E. & Zhu, Y. (2015). Identifying entities from scientific publications: A comparison of vocabulary- and model-based methods. *Journal of Informetrics*, 9(3), 455–465.
- Zhu, Y. & Yan, E. (2015). Dynamic subfield analysis of disciplines: An examination of the trading impact and knowledge diffusion patterns of computer science. *Scientometrics*, 104(1), 335-359.
- Kim, H., Zhu, Y., Kim, W., & Sun, T. (2014). Dynamic faceted navigation in decision making using Semantic Web technology. *Decision Support Systems*, 61, 59-68.