An Effective and Versatile Keyword Search Engine on Heterogenous Data Sources

Guoliang Li Jianhua Feng Jianyong Wang Lizhu Zhou
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China
{liguoliang,fengjh,jianyong,dcszlz}@tsinghua.edu.cn

ABSTRACT

We present EASE, an effective and versatile keyword search engine that enables users to easily access the heterogenous data composed of unstructured, semi-structured and structured data, without the need of learning XPath/XQuery or SQL languages. EASE addresses a challenge in keyword search that has been neglected in the literature: how to efficiently and adaptively process keyword queries on the heterogenous data. To provide such capability, EASE models unstructured, semi-structured and structured data as graphs, summarizes the graphs, and constructs graph indices instead of using traditional inverted indices for effective keyword search. EASE adopts an extended inverted index to facilitate keyword-based search, and employs a novel ranking mechanism for enhancing search effectiveness.

1. INTRODUCTION

Keyword search is a proven and widely popular mechanism for querying document systems and the World Wide Web. Recently, it has even been extensively applied to extract useful and relevant information from the Internet. Furthermore, the database research community has also recognized the benefits of keyword search and has been introducing keyword search capability into relational databases [1, 2, 5, 10, 11], XML databases [3, 6, 7, 8, 13], graph databases [4], and heterogenous data sources [9, 12].

Although many prior works of keyword search over textual documents (e.g., HTML documents) have been proposed, the existing web search engines (e.g., Google) always produce a list of individual pages as results and cannot integrate information from multiple interrelated pages to answer keyword queries meaningfully. In the event that there are no pages that contain all the keywords, they will return pages with some of the input keywords ranked by relevancy. Even if two or more interrelated pages contain all the keywords, the existing methods cannot integrate the pages into one relevant and meaningful answer. For example, to search for conferences covering the topic of "`Information Integration`" held in "Auckland" in 2008, one may issue a keyword query of "`Conference 2008 Auckland Information Integration`" to a search engine such as Google. As we all know, the venue of `VLDB 2008` is "`Auckland`" and "`Information Integration`" is one of its major research topics. Yet surprisingly, the homepage of `VLDB 2008` is neither in the top 10 results nor even in the first 100 answers. Why? The reason is that `VLDB 2008` splits its information into several pages methodically. The page of IMPORTANT DATE contains the keywords "`2008, Conference`" while "`Information Integration`" is contained in the CALL FOR PAPER page. Such data lineage problem also persists in most recently proposed community information management platforms, DBLife*.

Accordingly, the next-generation web search engines require *link-awareness*, or more generally, the capability of integrating correlative information items that are linked through hyperlinks. Meanwhile, the efficiency of keyword search on structured and semi-structured data remains a challenging problem. This is so because the traditional approaches have always employed the inverted index to process keyword queries, which is effective for unstructured data but inefficient for semi-structured and structured data. As the inverted index is inadequate for identifying the "`best`" answers with complex structural information, which is rather rich in XML documents or relational databases.

To the best of our knowledge, very few existing works could be universally applied to unstructured data (e.g., textual documents), semi-structured data (e.g., XML documents), structured data (e.g., relational databases) and graph data. Therefore, providing both effective and efficient search ability over such heterogeneous collections within a single search engine remains a big challenge. As it is, the structure of the data, such as the potentially hierarchical embedding in XML documents, is not fully exploited for answering keyword queries. It is also not taken into account for result ranking in most search engines. Consequently, current implementations focus on either IR-style search to meaningfully rank the results but ignore the rich structural information, or DB-style search to discover answers by identifying structural relationships but employ a very straightforward ranking mechanism.

This less-than-ideal situation calls for a framework for indexing and querying over large collections of unstructured, semi-structured or structured data, and adaptive ranking of the results retrieved over those heterogeneous data. In this demonstration, we propose EASE, an Efficient and Adaptive keyword Search Engine, to address these problems. EASE

*http://dblife.cs.wisc.edu/

PVLDB '08, August 23-28, 2008, Auckland, New Zealand
Copyright 2008 VLDB Endowment, ACM 978-1-60558-306-8/08/08

is in line with the current trend of seamlessly integrating databases (DB) and information retrieval (IR) techniques. EASE seamlessly integrates efficient query evaluation and adaptive scoring for ranking results. From the DB point of view, EASE provides an efficient algorithmic basis for scalable *top-k*-style processing of large amounts of heterogenous data for the discovery of rich structural relationships. EASE works by employing an adaptive, efficient and novel graph index beyond the inverted index. From the IR viewpoint, EASE integrates an effective ranking mechanism to improve search effectiveness.

To summarize, EASE has several salient features compared with existing keyword search engines:

- To the best of our knowledge, EASE is the first search engine that efficiently and versatilely processes keyword queries on heterogenous data sources composed of unstructured, semi-structured and structured data.
- EASE employs an effective graph index as opposed to the inverted index to improve search efficiency.
- EASE adopts a novel ranking mechanism for effective keyword search by taking into account both the structural compactness of answers from the DB viewpoint and the textual relevancy from the IR point of view.
- EASE uses a simple and yet efficient indexing mechanism to index the structural relationships between the transformed data. The index is amenable to the deployment of existing *top-k* ranking methods.
- Empirical evaluation shows that EASE generates search results with significantly improved precision and recall compared with the existing approaches. EASE also achieves high search efficiency, and outperforms existing state-of-the-art methods significantly.

2. EASE

2.1 r-Radius Steiner Graph

EASE models unstructured data (e.g., textual documents), semi-structured data (e.g., XML databases) and structured data (e.g., relational databases) as graphs, where the nodes are respectively documents, elements and tuples, and the edges are respectively hyperlinks, parent-child relationships (or IDREFS) and primary-foreign-key relationships.

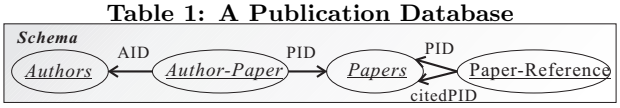
Inspired by the Steiner tree problem [2], which finds the Steiner trees from the database graph, EASE introduces *Steiner graph problem*, which identifies Steiner graphs as answers. Different from Steiner trees, Steiner graphs preserve much more complex structural information, such as circles.

As Steiner graphs with a larger radius[†] are not so meaningful and relevant to queries as users are generally frustrated by large and complex graphs, EASE introduces the concept of *r-radius graph*, which is the subgraph with radius no larger than a given threshold r.

Given an r-radius graph $\mathcal{G}$ and a keyword query $\mathcal{K}$. A node in $\mathcal{G}$ is called a *content node* if it directly contains some input keywords in $\mathcal{K}$. Node s in $\mathcal{G}$ is called a *Steiner node* if there exist two content nodes, u and v, and s is on the path between u and v (s may be u or v). The subgraph of $\mathcal{G}$ composed of the Steiner nodes and associated edges is called an *r-radius Steiner graph*. Accordingly, EASE addresses the problem of identifying the r-radius Steiner graphs.

[†]The radius of a graph is the minimum value of $r(u)$ for any node u, where $r(u)$ is the maximal distance from u to any other node.

Table 1: A Publication Database



Authors

AID	Name
a1	J. Shanmugasundaram
a2	L. Guo
a3	V. Hristidis
a4	Y. Papakonstantinou
a5	A. Balmin

Paper-Reference

PID	citedPID
p1	p2
p2	p3
p3	p4
p4	p5
p5	p6

Author-Paper

AID	PID
a1	p1
a1	p2
a2	p1
a3	p4
a3	p5
a4	p5
a4	p6
a4	p7
a5	p6

Papers

PID	Title
p1	Topology Search over Biological Databases
p2	XRANK: Ranked Keyword Search over XML Documents
p3	Bidirectional Expansion for Keyword Search on Graphs
p4	Finding *top-k* Answers in Keyword Proximity Search
p5	Efficient IR-Style Keyword Search over Relational Databases
p6	Keyword Proximity Search on XML Graphs
p7	DISCOVER: Keyword Search in Relational Databases

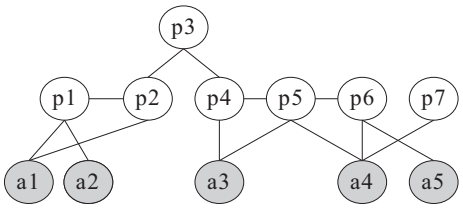


Figure 1: The Graph Model for the Publication Database in Table 1

For example, consider the database in Table 1. We model it to a graph $\mathcal{G}$ as illustrated in Figure 1. Given a query "`IR,Hristidis`" on the database in Table 1, we compute the Steiner graph composed of $p4$,$p5$ and $a3$ with associated edges between them as an answer. This differs from the Steiner tree (i.e., $p5-a3$) of prior studies, which can cause the loss of meaningful information, especially in databases with complicated structures.

Accordingly, we can take the r-radius Steiner graphs that contain all or a portion of the input keywords as answers, as the r-radius Steiner graphs are very compact and meaningful. Moreover, EASE will identify the *top-k* highest ranked r-radius Steiner graphs as the answer.

It is a big challenge to identify the r-radius Steiner graphs in a large graph. The efficiency and advantages of using inverted indices for facilitating the computation of the "`best`" answers for online keyword queries are well recognized. However, the inverted indices are not effective for discovering the much richer structural relationships existing in databases with complicated structures. It is therefore important to be able to efficiently and effectively discover these structural relationships, and index them for fast and accurate response. Intuitively, a straightforward way is to enumerate all the combinations of keywords, compute the corresponding r-radius Steiner graphs for each combination, and index these graphs. However, it is prohibitively expensive to dis-

cover all these structures since the number of combinations of all keywords in real databases is very large.

As an alternative, we propose an effective strategy to discover a portion of the r-radius graphs such that the number of which is proportional to the number of nodes in the graph, and we only need to index and materialize these graphs. More importantly, all of the r-radius graphs can be effectively identified through the indexed ones. Moreover, the r-radius Steiner graphs can be efficiently extracted on the fly. Interested readers refer to [12] for more details.

To facilitate efficient retrieval of r-radius graphs, we construct a novel *graph index*. The entries of the graph index are terms contained in the graph and each entry preserves the r-radius graphs that contain the term. To construct the graph index, we first extract r-radius graphs, then for each term k_i, we keep the set of all r-radius graphs that contain k_i, denoted as $\mathcal{I}_{k_i}$, i.e., $\mathcal{I}_{k_i}$=$\{\mathcal{G}^r|\mathcal{G}^r$ contains k_i, where $\mathcal{G}^r$ is an indexed r radius graph.$\}$.

To process a keyword query $\mathcal{K}$=$\{k_1,k_2,...,k_m\}$, we first retrieve the set $\mathcal{I}_{k_i}$ of those r-radius graphs which contain k_i based on the graph index, and then union every $\mathcal{I}_{k_i}$ to compute $\cup_{i=1}^{m}\mathcal{I}_{k_i}$, which is the set of r-radius graphs that contain all or a portion of the keywords in $\mathcal{K}$[‡]. Finally, we extract the r-radius Steiner graphs by removing the non-Steiner nodes from the corresponding r-radius graphs, and rank the results to return the *top-k* answers.

2.2 Ranking

The basic idea of the ranking method used in the existing literature is to first assign each r-radius graph a score using a standard IR-ranking formula (or its variants), and then combine the individual scores by using a score aggregation function (e.g., `SUM`) to obtain the final score. For example, the TF·IDF-based IR-style ranking function weights an r-radius Steiner graph by considering textual relevancy in IR literature, which takes into account term frequency (`tf`), inverse document frequency (`idf`) and normalized document length (`ndl`). `tf` and `idf` are well employed to rank documents in the IR literature while `ndl` is used to normalize document length as a longer document has a higher likelihood to contain many more keywords. Equation 1 gives the TF·IDF-based IR-style ranking function.

$$\textsc{Score}_{\textsc{IR}}(k_i,\mathcal{SG}) = \frac{tf_{(k_i,\mathcal{SG})} * idf_{k_i}}{ndl_{\mathcal{SG}}} \quad (1)$$

Although the TF·IDF-based ranking methods are efficient for textual documents, they are inefficient for semi-structured and structured data. From the IR perspective, traditional textual relevancy is important. However, due to our use of graph in modeling, the ranking of graph data becomes equally if not more important, and the structural compactness of r-radius Steiner graphs is the essence of the comparison. This is so because identifying rich structural relationships should be at least as important as discovering more keywords, and in some cases, even more crucial. Therefore, we propose a novel ranking function by incorporating structural compactness from the DB point of view.

Intuitively, when an r-radius Steiner graph $\mathcal{SG}$ is more compact, $\mathcal{SG}$ is more likely to be meaningful and relevant. Accordingly, the structural compactness score should be larger. As such, the compactness of $\mathcal{SG}$ should include the following parameters: i) the structural compactness between content nodes in $\mathcal{SG}$, and ii) the structural relevancy between input keywords w.r.t. $\mathcal{SG}$. We note that when the length of a path between two content nodes is larger, the relevancy between them is smaller. Further, there may be multiple paths between two content nodes, and we should consider all of them. Based on the above rationale, we propose Equation 2 to score the overall structural compactness between any two content nodes:

$$\textsc{Sim}(n_i,n_j) = \sum_{n_i \leftrightsquigarrow n_j} \frac{1}{(|n_i \leftrightsquigarrow n_j|+1)^2} \quad (2)$$

where $n_i \leftrightsquigarrow n_j$ denotes the path between n_i and n_j and $|n_i \leftrightsquigarrow n_j|$ denotes the number of nodes in the path.

Although the structural compactness between two content nodes can measure the structural relevancy of r-radius graphs, it cannot evaluate the structural relevancy among input keywords, which captures the phrase-based relevancy between input keywords. It follows that a smaller distance between input keywords indicates a higher structural relevancy between them. This is particularly so for keywords in the same node that will represent a phrase. We therefore propose Equation 3 to capture this parameter.

$$\textsc{Sim}(< k_i,k_j > |\mathcal{SG}) = \frac{1}{|\mathcal{C}_{k_i} \cup \mathcal{C}_{k_j}|} \sum_{n_i \in \mathcal{C}_{k_i};\ n_j \in \mathcal{C}_{k_j}} \textsc{Sim}(n_i,n_j) \quad (3)$$

where $\mathcal{C}_{k_i}$ denotes the set of all the content nodes that contain k_i in $\mathcal{SG}$, and $|\mathcal{C}_{k_i}|$ denotes the number of nodes in $\mathcal{C}_{k_i}$, which is used to normalize the structural relevancy between two input keywords. Consequently, a larger overall structural compactness score of $\mathcal{SG}$ indicates that $\mathcal{SG}$ is more likely to be relevant and meaningful to $\mathcal{K}$.

By taking into account both document relevancy from the IR perspective and structural compactness/relevancy from the DB perspective to capture structural relationships, we present a more accurate function for scoring r-radius Steiner graphs as given in Equation 4.

$$\textsc{Score}(\mathcal{K},\mathcal{SG}) = \sum_{1 \le i < j \le m} \textsc{Score}(< k_i,k_j > |\mathcal{SG}) \quad (4)$$

where

$$\textsc{Score}(< k_i,k_j > |\mathcal{SG}) = \textsc{Sim}(< k_i,k_j > |\mathcal{SG}) * (\textsc{Score}_{\textsc{IR}}(k_i,\mathcal{SG}) + \textsc{Score}_{\textsc{IR}}(k_j,\mathcal{SG})) \quad (5)$$

$\textsc{Score}(<k_i,k_j>|\mathcal{SG})$ measures the overall relevancy score of $<k_i,k_j>$ in $\mathcal{SG}$ based on the structural compactness/relevancy and IR scores. Note that, $\textsc{Sim}(<k_i,k_j>|\mathcal{SG})$ is taken as the weight of the sum of two IR scores, i.e., $\textsc{Score}_{\textsc{IR}}(k_i,\mathcal{SG})$ and $\textsc{Score}_{\textsc{IR}}(k_j,\mathcal{SG})$. A larger $\textsc{Sim}(<k_i,k_j>|\mathcal{SG})$ means that k_i and k_j are more relevant w.r.t. $\mathcal{SG}$, and thus, the overall score of $<k_i,k_j>$ in $\mathcal{SG}$ is expected to be larger.

2.3 Indexing

EASE also employs an extended inverted index (EI-Index) for facilitating keyword-based search.

Given any two keywords k_i and k_j in the graph, and an r-radius graph $\mathcal{SG}$, the scores of $\textsc{Score}_{\textsc{IR}}(k_i,\mathcal{SG})$ and $\textsc{Score}_{\textsc{IR}}(k_j,\mathcal{SG})$ in Equation 1 and $\textsc{Sim}(<k_i,k_j>|\mathcal{SG})$ in Equation 3 share the key feature that they can be pre-computed and materialized off-line. Based on this observation, we can materialize $\textsc{Score}(<k_i,k_j>|\mathcal{SG})$ into EI-Index.

[‡]If we consider "AND" predicate, we merge each $\mathcal{I}_{k_i}$ to compute $\cap_{i=1}^{m}\mathcal{I}_{k_i}$, which is the set of r-radius graphs that contain all keywords.

Table 2: EI-Index: An Extended Inverted Index

Paired-Keywords	<r-radius Graph, Score>
<Database, DISCOVER>	$\mathcal{G}^r_{p_5}$,1.53
<Database, Papakonstantinou>	$\mathcal{G}^r_{p_5}$,0.38; $\mathcal{G}^r_{p_4}$,0.19
<Database, Relational>	$\mathcal{G}^r_{p_5}$,0.85; $\mathcal{G}^r_{p_3}$,0.35; $\mathcal{G}^r_{p_4}$,0.34
<DISCOVER, Papakonstantinou>	$\mathcal{G}^r_{p_5}$,0.54
<DISCOVER, Relational>	$\mathcal{G}^r_{p_5}$,1.95
<Papakonstantinou, Relational>	$\mathcal{G}^r_{p_5}$,0.57; $\mathcal{G}^r_{p_4}$,0.28
<$\cdots$, $\cdots$>	$\cdots$

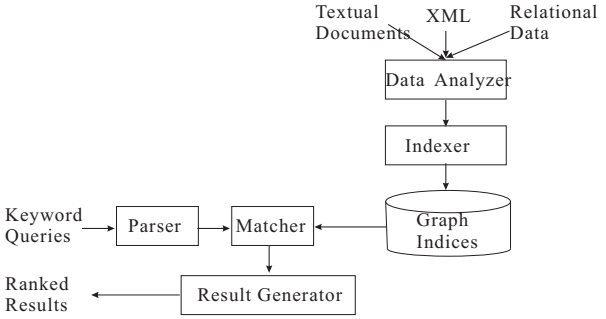


Figure 2: The Architecture of EASE

Different from the traditional inverted index, the entries of EI-Index are paired-keywords (combinations of two keywords), and the value of each entry is the r-radius graphs that contain the paired-keywords and the corresponding scores. For example, we can construct the EI-Index of the graph in Figure 1 as illustrated in Table 2.

To answer a keyword query $\mathcal{K}$=$\{k_1,k_2,\cdots,k_m\}$, we first retrieve the r-radius graphs for every paired-keywords $<k_i, k_j>$ according to EI-Index, and then compute the scores of every relevant r-radius graph according to Equation 4. Finally, we rank the results and return the *top-k* r-radius Steiner graphs with the highest scores by refining the corresponding r-radius graphs. Moreover, we introduce an effective technique of progressively identifying the *top-k* answers. We employ threshold based techniques [12] to identify the *top-k* answers on top of our EI-Index.

3. IMPLEMENTATION

The system architecture of EASE is illustrated in Figure 2. The `Data Analyzer` parses the input unstructured, semi-structured and structured data, and models them as graphs. The `Indexer` incrementally constructs the graph indices. Once a user issues a query, `Parser` parses it and `Matcher` accesses the indices to retrieve the r-radius graphs that match the query. Finally, `Result Generator` outputs the ranked *top-k* search results by refining the r-radius graphs.

We have implemented EASE in Java. It takes user input keywords and textual documents(e.g, HTML documents, PDF, WORD, PPT, etc.), XML documents, and relational data as input, and returns results that match the input keyword queries. The empirical study of EASE in comparison with existing methods has been presented in [12]. Experiments show that EASE significantly outperforms state-of-the-art methods in search quality measured by precision, recall and F-measure, as well as search efficiency.

4. DEMONSTRATION

What is the goal of the demo? Through the demo, we present a challenge of keyword search over heterogenous data: how to effectively and versatilely answer keyword queries on unstructured, semi-structured and structured data. The development and demonstration of EASE show the following promising features: *i*) EASE provides an effective and adaptive search engine for heterogenous data; *ii*) EASE improves the search efficiency by adopting the novel graph index; *iii*) EASE improves the search quality by employing a novel ranking mechanism which seamlessly integrates the structural compactness of answers from DB viewpoint and the textual relevancy from IR point of view.

What will be shown in the demo? EASE provides a web-based interface (http://dbgroup.cs.tsinghua.edu.cn/EASE/) which allows users to specify data and keyword queries for retrieval. Various sample textual documents (e.g., DBLife), XML documents (e.g., DBLP), relational data (e.g., IMDB), and the mixed heterogenous data are provided. Moreover, users can select different values of k to retrieve the *top-k* relevant results. In the demonstration, besides demonstrating the functionalities of EASE, we present the design and architecture we employ in developing the system.

5. ACKNOWLEDGEMENT

Sincere thanks go to Professor Beng Chin Ooi for his insightful guidance and earnest help all through the analysis and paper-writing stages.

This work is partly supported by the National Natural Science Foundation of China under Grant No.60573094, the National High Technology Development 863 Program of China under Grant No.2007AA01Z152 and 2006AA01A101, the National Grand Fundamental Research 973 Program of China under Grant No.2006CB303103, and Basic Research Foundation of Tsinghua National Laboratory for Information Science and Technology (TNList).

6. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[2] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using banks. In *ICDE*, pages 431–440, 2002.

[3] L. Guo, F. Shao, C. Botev, and J. Shanmugasundaram. Xrank: Ranked keyword search over xml documents. In *SIGMOD*, pages 16–27, 2003.

[4] H. He, H. Wang, J. Yang, and P. Yu. Blinks: Ranked keyword searches on graphs. In *SIGMOD*, 2007.

[5] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient ir-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[6] G. Li, J. Feng, J. Wang, and L. Zhou. Efficient keyword search for valuable lcas over xml documents. In *CIKM*, 2007.

[7] G. Li, J. Feng, J. Wang, and L. Zhou. Efficient keyword search over data-centric xml documents. In *APweb*, 2007.

[8] G. Li, J. Feng, J. Wang, and L. Zhou. Race: Finding and ranking compact connected trees for keyword proximity search over xml documents. In *WWW*, 2008.

[9] G. Li, J. Feng, J. Wang, and L. Zhou. Sailer: An effective search engine for unified retrieval of heterogeneous xml and web documents. In *WWW*, 2008.

[10] G. Li, J. Feng, and L. Zhou. Finding dominate trees for effective keyword search over relational databases. In *BNCOD*, 2008.

[11] G. Li, J. Feng, and L. Zhou. RETUNE: Retrieving and Materializing Tuple Units for Effective Keyword Search over Relational Databases. In *ER*, 2008.

[12] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. Ease: Efficient and adaptive keyword search on unstructured, semi-structured and structured data. In *SIGMOD*, 2008.

[13] Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest lcas in xml databases. In *SIGMOD*, pages 527–538, 2005.