Conjunctive Keyword Search with Designated Tester and Kjerhyretiming Enabled Proxy Re-Encryption

B.Sravanthi
PG Scholar
K.N.MadhaviLatha
Assitant Professor
Dept of CSE, Sir C RReddy College of Engineering, Eluru, Andhrapradesh, India.

Abstract-Instant search is an emerging information-retrieval paradigm in which a system finds answers to a query instantly while a user types in keywords character-by-character. Fuzzy search further improves user search experiences by finding relevant answers with keywords similar to query keywords. A main computational challenge in this paradigm is the high speed requirement, i.e., each query needs to be answered within milliseconds to achieve an instant response and a high query throughput. At the same time, we also need good ranking functions that consider the proximity of keywords to compute relevance scores. In this paper, we study how to integrate proximity information into ranking in instant-fuzzy search while achieving efficient time and space complexities. We study how to index these phrases and develop an incremental-computation algorithm for efficiently segmenting a query into phrases and computing relevant answers. We conducted a thorough experimental study on real data sets to show the tradeoffs between time, space, and quality of these solutions. Our results indicate that many existing search techniques do not provide acceptable performance for realistic retrieval tasks. In particular, memory consumption precludes many search techniques from scaling beyond small data sets with tens of thousands of vertices. We also explore the relationship between execution time and factors varied in previous evaluations; our analysis indicates that most of these factors have relatively little impact on performance. In summary, our work confirms previous claims regarding the unacceptable performance of these search techniques and underscores the need for standardization in evaluations—standardization exemplified by the IR communit

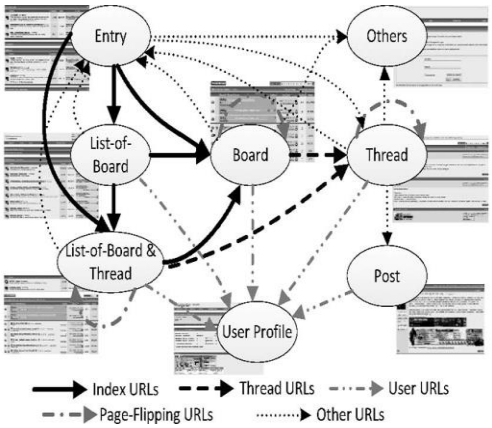
Index Terms—Keyword search, relational database, information retrieval, empirical evaluation

I. INTRODUCTION

In previous work we proposed the first benchmark to evaluate relational keyword search techniques and evaluated them with regard to their search effectiveness. However, our previous work did not consider the runtime performance of these search techniques, which is our focus in this paper. Unlike many evaluations that appear in the literature, our benchmark uses realistic data sets and realistic queries to investigate the numerous tradeoffs made in the design of these search techniques. Our benchmark is the only one to date in the literature that satisfies the minimum criteria established by the IR community for the evaluation of retrieval systems. The major contributions of this paper are as follows: We conduct an independent, empirical evaluation of the runtime performance of seven relational keyword search techniques. Our evaluation is the most extensive and thorough one to appear to date in the literature. Our results do not substantiate previous claims regarding the scalability and performance of relational keyword search techniques. Existing search techniques perform poorly on databases exceeding tens of thousands of tuples or require an inordinate amount of memory. We show that many parameters varied in existing evaluations are at best loosely correlated with runtime performance. The lack of a meaningful relationship gives merit to previous claims of unpredictable performance for existing search techniques.Our work is the first to combine performance and search effectiveness in the evaluation of such a large number of search techniques. Considering these two issues in conjunction provides better understanding of these two critical tradeoffs among competing approaches. The remainder of this paper is organized as follows: Section 2 formally defines the problem of keyword search in relational data graphs and describes the search techniques included in our evaluation. Section 3 describes our experimental setup, including our evaluation benchmark and metrics. In Section 4, we present our experimental results, and we discuss them in Section 5. We review related work in Section 6 and provide our conclusions in Section 7. Online appendices provide greater detail about our evaluation benchmark and summarize implementation details of the search techniques. We use two metrics to measure runtime performance. The first is execution time, which is the time elapsed from issuing a query until an algorithm terminates. Because there are a large number of potential results for each query, search techniques typically return only the top-k results where k specifies the desired retrieval depth. Our second metric is response time, which we define as the timeelapsed from issuing the query until i results have been returned (where i < k). Because this

definition is not welldefined when fewer than k results are retrieved, we define it for j, where i < j < k and i is the number of resultsretrieved and k is the desired retrieval depth, as the algorithm's execution time. Effectiveness metrics are also critical to the evaluation of retrieval systems because not every result is actually relevant to the query's underlying information need. Recallis the ratio of relevant results retrieved to the total number of relevant results. Precision is the ratio of relevant results retrieved to the total number of retrieved results. Precision @ k (P@k) is the mean precision across multiple querieswhere the retrieval depth is limited to k results. If fewer than k results are retrieved by a system, we calculate the precision value at the last result. We also use MAP to measure retrieval effectiveness at greater retrieval depths. Measuring the completeness of the set of of search results returned by a particular search technique is tempting, but only Golenberg et al.'s algorithm is proven to becomplete (i.e., return all possible results) for the given search terms. Furthermore, it is not clear what effect omitting some results may have on a search technique. Unlike recall, which is measured against the set of relevant results, omitting a few results may have practically no impact on the effectivenessof the search technique, particularly if the omitted results are highly redundant with others that are enumerated. More importantly, there is no precedent from the IR community to evaluate retrieval systems using a purelyobjective metric because retrieval systems explicitly answer subjective information needs. We implemented BANKS, DISCOVER, DISCOVER-II, and DPBF and obtained implementations of BANKS-II (i.e., the bidirectional search algorithm), BLINKS, and STAR. All the search techniques are implemented in Java. For some search techniques, we also had access to others' implementations. Among the implementations, we found that our reimplementations generally outperform the implementations provided by others. Exceptions to this trend were the result of correcting significant implementation defects. Our experiments do not compare against traditional IR systems (e.g., Apache Lucene5) because more traditional systems do not consider the relationships among database tuples, which is an important aspect of relational keyword search. Our implementation of BANKS adheres to its original description although it queries the database dynamically to identify nodes (tuples) that contain query keywords. Our implementation of DISCOVER borrows its successor's query processing techniques. Both DISCOVER and DISCOVER- II are executed with the sparse algorithm, whichprovides the best performance for queries with AND semantics [17]. BLINKS's block index was created using breadth-first partitioning and contains 50 nodes per block.6 STAR uses the edge weighting scheme proposed by Dinget al. [12] for undirected graphs.

INTERNET forums (also called web forums) are important services where users can request and exchange information with others. For example, the Trip Advisor Travel Board is a place where people can ask and share travel tips. Due to the richness of information in forums, researchers are increasingly interested in mining knowledge from them. Zhai and Liu [28], Yang et al. [27], and Song et al. [23] extracted structured data from forums. Gao et al. [15] identified question and answer pairs in forum threads. Zhang et al. [30] proposed methods to extract and rank product features for opinion mining from forum posts. Glance et al. [16] tried to mine business intelligence from forum data. Zhang et al. [29] proposed algorithms to extract expertise network in forums. To harvest knowledge from forums, their content must be downloaded first. However, forum crawling is not a trivial problem. Generic crawlers [12], which adopt a breadth-first traversal strategy, are usually ineffective and inefficient for forum crawling. This is mainly due to two non crawler friendly characteristics of forums [13], [26]: 1) duplicate links and uninformative pages and 2) page-flipping links. A forum typically has many duplicate links that point to a common page but with different URLs [7], e.g., shortcut links pointing to the latest posts or URLs for user experience functions such as "view by date" or "view by title." A generic crawler that blindly follows these links will crawl many duplicate pages, making it inefficient. A forum also has many uninformative pages such as login control to protect user privacy or forum software specific FAQs. Following these links, a crawler will crawl many uninformative pages. Though there are standard-based methods such as specifying the "rel" attribute with the "no follow" value (i.e., "rel ¼ no follow") [6], Robots Exclusion Standard (robots.txt) [10], and Sitemap [9] [22] for forum operators to instruct web crawlers on how to crawl a site effectively, we found that over a set of nine test forums more than 47 percent of the pages crawled by a breadth-first crawler following these protocols were duplicates or uninformative. This number is a little higher than the 40 percent that Cai et al. [13] reported but both show the inefficiency of generic crawlers. More information about this testing can be found in bellow Section



Auto-Completion: In auto-completion, the system suggests several possible queries the user may type in next. There have been many studies on predicting queries many systems do prediction by treating a query with multiple keywords as a single prefix string. Therefore, if a related suggestion has the query keywords but not consecutively, then this suggestion cannot be found.

Instant Search: Many recent studies have been focused on instant search, also known as *type-ahead search*. The studies in proposed indexing and query techniques to support instant search. The studies in presented triebased techniques to tackle this problem. Studied instant search on relational data modeled as a graph.

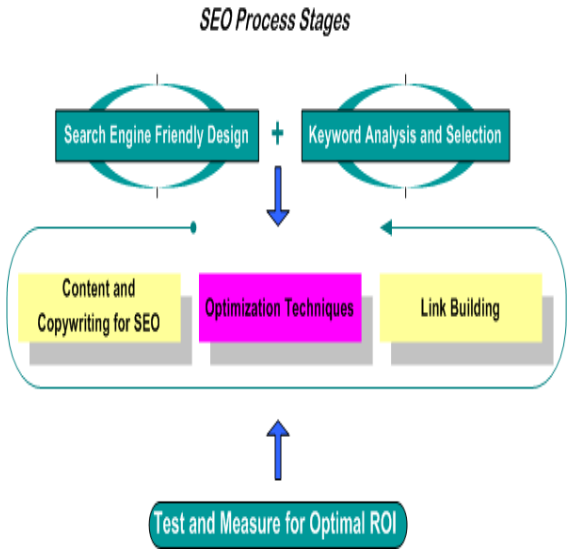
Fuzzy Search: The studies on fuzzy search can be classified into two categories, gram-based approaches and trie-based approaches. In the former approach, sub-strings of the data are used for fuzzy string matching. The second class of approaches index the keywords as a trie, and rely on a traversal on the trie to find similar keywords. This approach is especially suitable for instant and fuzzy search since each query is a prefix and trie can support incremental computation efficiently.

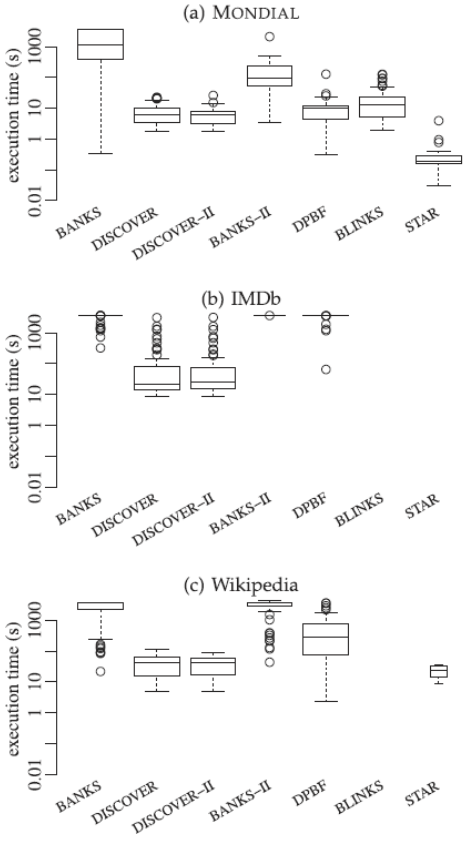
Proximity Ranking: Recent studies show proximity is highly correlated with document relevancy, and proximityaware ranking improves the precision of top results significantly. However, there are only a few studies that improve the query efficiency of proximity-aware search by using early-termination techniques exploited document structure to build a multi-tiered index to terminate the search process without processing all the tiers.

Admin Module: This module is used to help the server to view details and upload files Details. The admin after the login and view the user downloading details and the counting of file request details on flowchart.

Literature Survey: Existing methods only identify a single tuple unit to answer keyword queries. However, they neglect the fact that in many cases, we need to integrate multiple related tuple units to answer a keyword query. To address this problem, in this paper, we propose a structure-aware-index-based method to integrate multiple related tuple units to effectively answer keyword queries.

Proposal of the Proposed System: The proposed to improve search efficiency by indexing structural relationships, and existing methods identify a single tuple unit to answer keyword queries. However, in many cases, multiple tuple units should be integrated to answer a keyword query. Thus, these methods will involve false negatives. To address this problem, in this paper, we study how to integrate multiple related tuple units to effectively answer keyword queries. To achieve a high performance, we propose an approach that focuses on common phrases in the data and queries, assuming records with these phrases are ranked higher. We study how to index these phrases and develop an incremental-computation algorithm for efficiently segmenting a query into phrases and computing relevant answers. We conducted a thorough experimental study on real data sets to show the tradeoffs between time, space, and quality of thesesolutions.





II. CONCLUSION

Unlike many evaluations reported in the literature, ours investigates the overall, end-to-end performance of relational keyword search techniques. Hence, we favor a realistic query workload instead of a larger workload with queries that are unlikely to be representative (e.g., queries created by randomly selecting terms from the data set). Our experimental results do not reflect well on existing relational keyword search techniques. Runtime performance is unacceptable for most search techniques. Memory consumption is also excessive for many search techniques. Our experimental results question the scalability and improvementsclaimed by previous evaluations. These conclusions are consistent with previousevaluations that demonstrate the poor runtime performance of existing search techniques as a prelude to a newly-proposed approach.

III. PROPOSED SYSTEM

Further research is unquestionably necessary to investigate the myriad of experimental design decisions that have a significant impact on the evaluation of relational keyword search systems. For example, our results indicate that existing systems would be unable to search the entire IMDb database, which underscores the need for a progression of data sets that will allow researchers to make progress toward this objective. Creating a subset of the original data set is common, but we are not aware of any work that identifies how to determine if a subset is representative of the original data set. In addition, different research groups often have different schemas for the same data (e.g., IMDb),but the effect of different database schemas on experimental results has also not been studied. Our results should serve as a challenge to this community because little previous work has acknowledged these challenges. Moving forward, we must address several issues. First, we must design algorithms, data structures, and implementations that recognize that main memory is limited. Search techniques must manage their memory utilization efficiently, swapping data to and from disk as necessary. Such implementations are unlikely to have performance characteristics that are similar to existing approaches but must be used if relational keyword search systems are to scale to large data sets (e.g., hundreds of millions of tuples). Second, evaluations should reuse data sets and query workloads to provide greater consistency of results, for even our results vary widely depending on which data set is considered. Having the community Coalesce behind reusable test collections would facilitate better comparison among systems and improve their overall evaluation. Fortunately, our evaluation benchmark is beginning to gain traction in this area as evidenced by others' adoption of it for their evaluations. Third, the practice of researchers reimplementing search techniques may account for some evaluation discrepancies. Making the original source code (or a binary distribution that accepts a database URL and query as input) available to other researchers would greatly reduce the likelihood that observed differences are implementation artifacts.

IV. REFERENCES

[1]. G. Li, J. Feng, X. Zhou, and J. Wang, "Providing Built-in Keyword Search Capabilities in RDBMS," The VLDB J., vol. 20, pp. 1-19, Feb. 2011.
[2]. V. Hristidis and Y. Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases," Proc. 28th Int'l Conf. Very Large Data Base (VLDB '02), pp. 670-681, Aug. 2002.
[3]. V. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IRStyle Keyword Search over Relational Databases," Proc. 29th Int'l Conf. Very Large Data Bases (VLDB '03), pp. 850-861, Sept. 2003.
[4]. A. Singhal, J. Choi, D. Hindle, D. Lewis, and F. Pereira, "AT&T at TREC-7," Proc. Seventh Text REtrieval Conf. (TREC-7), pp. 239-252, Nov. 1999.
[5]. S.E. Dreyfus and R.A. Wagner, "The Steiner Problem in Graphs," Networks, vol. 1, no. 3, pp. 195-207, 1971.
[6]. G. Reich and P. Widmayer, "Beyond Steiner's Problem: A VLSI Oriented Generalization," Proc. 15th Int'l Workshop Graph-Theoretic Concepts in Computer Science, pp. 196-210, 1990.
[7]. W. May, "Information Extraction and Integration with Florid: The Mondial Case Study," Technical Report 131, Universita¨t Freiburg, Institut fu¨ r Informatik, 1999.
[8]. G. Pass, A. Chowdhury, and C. Torgeson, "A Picture of Search," Proc. First Int'l Conf. Scalable Information Systems (InfoScale '06), May 2006.
[9]. J. Coffman and A.C. Weaver, "What Are We Searching For? Analyzing User Objectives When Searching Relational Data," Proc. Workshop Web Search Click Data (WSCD '12), Feb. 2012.
[10]. K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword Proximity

Naga Madhavi Latha Kakarla received M.Tech in Computer Science from College of Engineering, Jawaharlal Nehru Technological University, Hyderabad, Andhra Pradesh, India. Presently, she is working as Assistant Professor in Sir C.R.Reddy College Of Engineering, Eluru, West Godavari District. Andhra Pradesh, India. She is currently doing her Ph.D from Acharya Nagarjuna University, Guntur. A.P, India. She has 13 years of Experience in Teaching. Her research interest includes Data mining and Bioinformatics.

B.Sravanthi Presently pursuing her M.Tech inComputer Science & Technology from Sir C R Reddy Engineering College, Vatluru, West Godavari District, A.P, India. Affiliated to Andhra University, Approved by AICTE, New Delhi. University Kakinada, West Godavari District, A.P, India.