International Journal of Technology and Engineering System (IJTES)
Vol 7. No.5 2015 Pp. 460-469

available at : www.ijcns.com
ISSN: 0976-1345

A Speculative Performance Appraisal of Comparative Keyword Search Scheme

Vijaya. A[1], Madhangopi. A, M.E.,(Ph.D)[2]
[1]ME(CSE) Student, [2]Assistant Professor/CSE
[1,2]Sri Muthukumaran Institute of Technology, Chennai.
[1]vijayakas28@ymail.com, [2]madhangopi@gmail.com

ABSTRACT

The keyword search paradigm to relational data has been an active area of research within the database and information retrieval (IR) community. Various methodologies are being proposed and implemented but despite numerous publications there remains a severe lack of standardization for system evaluations. This lack of standardization produces plenty of contradictory in results. This system provides standardization for the performance evaluation. It produces efficiency in execution time and memory consumption. This scheme uses update of the files with keyword search for the consumer. This analysis includes the BLINKS technique for providing hierarchy to the search and it uses the BANKS, DISCOVER to provide a ranking of the scheme. It gives the length of the file using bidirectional expansion for keyword search on graph databases. This scheme provides the solution for all structured and semi-structured data. This solution contains the size of the file using the technique of the compression. This system achieves various advantages related with the files of the data in keyword search.

Keywords: relational data, memory consumption, bidirectional expansion, information retrieval.

I. Introduction

The pervasive searching text box has changed the way public interact with data. Nearly fifty percent of the internet users use web search engine daily[10],performing in excess of more billion searches[11].The victory of keyword search scheme from what it does not require a specific query language or data knowledge of the underlying structure of the information. Web users increasingly asking keyword search intermediate for getting information and it is very normal to elaborating this paradigm for relational data. This elaboration has been an active area of research in the past decade. We posit the information that the existing and their evaluations performed by the various researchers are not indicative of the these schemes in the real time performance.

Even though important number of research papers being implemented in this area, existing speculative evaluations ignore or only partially explain many important issues related to search performance. Bhalotia [2] says that existing evaluation of the system performance would be unpredictable. This will explain about the important of this real world tasks. This will gain little support in the existing literature, but the failure of these systems to gain a foothold implies that robust ,independent evaluation is necessary. In part, existing performance issues may be concealed by experimental design decisions

such as the option of datasets or the building of query workloads. Therefore we conduct not dependent ,speculative evaluation of previous comparative keyword search techniques using an openly available benchmark to ascertain their real-world performance for realistic query workloads.

i.Keyword searching:- Keyword searching is the process of of searching electronically stored evidence using any specified word, or combination of words with the intent of locating and identifying potential evidence. The process involves careful planning and review of keywords so that only the relevant documents are produced.

ii.Keyword Analysis:-We describe BANKS[2], a system which enables keyword-based search on relational databases, together with data and schema browsing. BANKS[2] enables users to extract information in a simple manner without any knowledge of the schema or any need for writing complex queries. A user can get information by typing a few keywords, following hyperlinks, and interacting with controls on the displayed results.

iii.Keyword Querying and Ranking:-One approach that has been explored is to allow users to query such databases in the same ways as they explore web documents. Thus, it is desirable to be able to use the paradigm of keyword querying and automated result ranking over contents of databases. However, the rich relationships and schema information present in databases makes a direct adaptation of information retrieval techniques inappropriate.

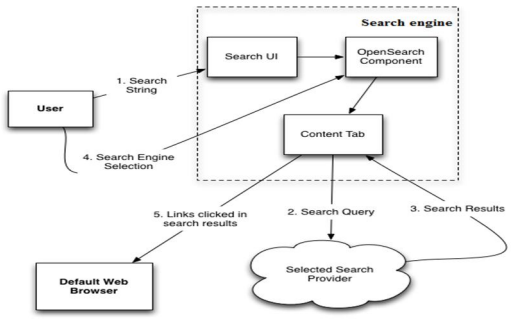

Fig.1. Keyword Search Scheme

A. Comparative Keyword Search Scheme

Keyword search on semi-structured data(e.g.HTML,XML) and comparative or relational data differs considerably from traditional IR. A divergence exists between the data's physical storage and logical view of the information. Comparative or Relational databases are normalized to neglect redundancy and primary keys identify unique information. Search queries frequently cross these relationships that is subset of search terms evaluated first then the related tuples found automatically, which forces comparative keyword search scheme to recover a logical view of the information. The internal assumption of keyword search that is search terms are related and the process will be complicated. It is almost possible to add another search term with existing result. This realization leads to pressure between the comfort and coverage of search results.

B. Contributions

As we discuss previous in this paper, many relational keyword search scheme approximate solutions to intractable problems. Researchers consequently rely on speculative evaluation to validate their heuristics. we continue this tradition by evaluating these systems using a benchmark designed for comparative or relational keyword search. On the view of retrieval process exposes the real-world trade-offs made in the design of many of these systems. But, some systems use alternative methods to improve performance compare with other methods. These are all not focus on prior evaluations.

The major contributions of this paper as follows:

- We conduct an independent ,speculative performance evaluation of 5 relational or comparative keyword search scheme which reduces the work as comparison with previous.
- Keyword search uses the ranking system. Which filter the file based on their file size and order of usage in the database and user.
- Program execution takes the less time and the memory consumption very less compare to other techniques. In this technique very much efficient.
- Length of the file can be seen by every user of the application. The execution time also shown for everyone in the viewable type of the task.

- File ranking can be seen by everyone in the usage of chart. It will give clear identification of the project.
- It is compatible with every system and it will show the range of the information should be unique with other technique.
- It is efficient with other technique and it is used for the all user .

The remainder of this paper is organized as follows. In part II, we motivate this work by describing previous evaluations and why an independent evaluation of these systems is warranted. Part III formally defines the problem of keyword search in relational data graphs and describes the schemes included in our evaluation. Part IV explains our experimental setup, including our evaluation benchmark and metrics. In Part V, we explain our experimental results, including possible threats to validity. We review related work in Part VI and provide our conclusions in Part VII.

II. Objective for Independent Evaluation

Most evaluations in the literature disagree about the performance of various search techniques, but important experimental design differences may account for these difficulties. We discuss three such differences in this section.

A. Datasets

Figure 2 summarizes the datasets and the number of queries used in previous evaluations. Although this graph suggests some uniformity in evaluation datasets, their content varies dramatically. Consider the evaluations of BANKS-II [17], BLINKS [13], and STAR [18]. Only BANKS-II's evaluation includes the entire Digital Bibliography & Library Project (DBLP) and the Internet Movie Database (IMDb) . Both BLINKS and STAR use smaller subsets to facilitate comparison with systems that assume the data graph fits entirely within main memory. The literature does not address the representativeness of database subsets, which is a serious threat because the choice of a subset has a profound effect on the experimental results. For example, a subset containing one percent of the original data is two orders of magnitude easier to search than the original database due to fewer tuples containing search terms.

B. Query Workloads

The query workload is another critical factor in the evaluation of these systems. The trend is for researchers either to create their own queries or to create queries from terms selected randomly from the corpus. The latter strategy is particularly poor because queries created from randomly-selected terms are unlikely to resemble real user queries [23]. The number of queries used to evaluate these systems is also insufficient. The traditional minimum for evaluating retrieval systems is 50 queries [32] and significantly more may be required to achieve statistical significance [34]. Only two evaluations that use relatistic query workloads meet this minimum number of information needs.

C. Testing Difficulties

Discrepancies among existing evaluations are prevalent. Table II lists the mean execution times of systems from three evaluations that use DBLP and IMDb databases. The table rows are search techniques; the columns are different evaluations of these techniques. Empty cells indicate that the system was not included in that evaluation. According to its authors, BANKS-II "significantly outperforms" [17] BANKS, which is supported by BANKS-II's evaluation, but the most recent evaluation contradicts this claim especially on DBLP. Likewise, BLINKS claims to outperform BANKS-II "by at least an order of magnitude in most cases" [13], but when evaluated by other researchers, this statement does not hold. We use Table II to motivate two concerns that we have regarding existing evaluations. First, the difference in the relative performance of each system is startling. We do not expect the most recent evaluation to downgrade the orders of magnitude performance improvements to performance degradations, which is the certainly the case on the DBLP dataset. Second, the absolute execution times for the search techniques vary widely across different evaluations. The original evaluation of each system claims to provide "interactive" response times (on the order of a few seconds) but other evaluations strongly refute this claim.

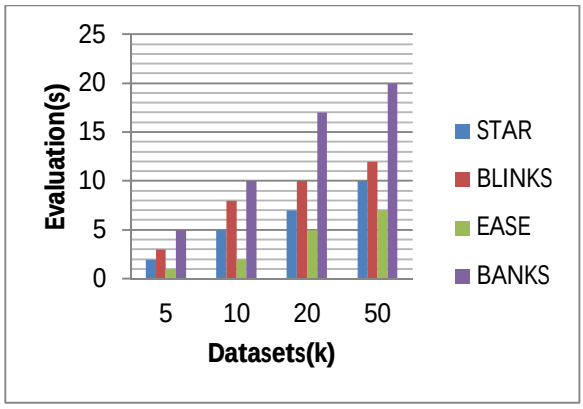

Fig 2.Statistics from evaluation

III. Relational Keyword Search Systems

In this part explains about the speculative evaluation of the system, we took general model of keyword search over data graphs. It includes the search techniques in our evaluation. Other evaluation techniques mentioned in Part VI.

A. Query based Schemes

Query based approaches support keyword search over relational databases directly executes SQL commands. These techniques model the relational schema as a graph where edges denote relationships between tables. The database's full text indices all tuples contain search terms and a join expression is created for each possible relationship between these tuples. DISCOVER [15] creates a set of tuples for each subset of search terms in the database relations. A candidate network is a tree of tuple sets where edges correspond to relationships in the database schema. DISCOVER enumerates candidate networks using a breadth-first algorithm but limits the maximum size to ensure efficient enumeration. A smaller size improves performance but risks missing results. DISCOVER creates a join expression for each candidate network, executes the join expression against the underlying database to identify results, and ranks these results by the number of joins. Hristidis et al. [14] refined DISCOVER by adopting pivoted normalization scoring [30] to rank results.

B. Graph based schemes

The aim of nearest search is to minimize the weight of the result trees. This work is a formulation of the group of Steiner tree problem[9],which is known to be NP-complete[29].Graph based searches are more general than query based approaches, for relational databases, XML and Internet can all be modelled as graphs.BANKS[2] produces the results by searching the graph backwards from vertices that contain query overload keywords. The backward search technique concurrently executes copies of Dijkstra's shortest path algorithm[7],one from each vertex that contains a search term. When a vertex has been labelled with its distance to each search term, that vertex is the root of a directed tree that is a result of the query.

BANKS-II[17] elements the backward search technique[2] by searching the graph forwards from potential root nodes. This strategy has an advantage when the query contains a common term or when a copy of Dijkstra's shortest path algorithm reaches a vertex with a large number of incoming edges. Spreading activation prioritizes the search but may cause the bidirectional search technique to identify shorter path is found, the existing results must be updated recursively, which potentially increases the total execution time.

IV. Appraisal Framework

In this part, we present our appraisal framework. We start by describing the benchmark [5] that we use to evaluate the various keyword search techniques. We then describe the metrics we report for our experiments and our experimental setup.

A. Criterion Overview

Our evaluation benchmark includes the three datasets such that MONDIAL [24], IMDb, and Wikipedia. Two datasets (IMDb and Wikipedia) are extracted from popular websites. The size of the datasets varies widely: MONDIAL is more than two orders of magnitude smaller than the IMDb dataset, and Wikipedia lies in between. In addition, the schemas and content also differ considerably. MONDIAL has a complex schema with almost 30 relations while the IMDb subset has only 6. Wikipedia also has few relations, but it contains the full text of articles, which emphasizes more complex ranking schemes for results. Our datasets roughly span the range of dataset sizes that have been used in other evaluations.

The benchmark's query workload was constructed by researchers and comprises 50 information needs for each dataset. The query workload does not use real user queries extracted from a search engine log for three reasons. First, Internet search engine logs do not contain queries for datasets not derived from websites. Second, many queries are inherently ambiguous and knowing the user's original information need is essential for accurate relevance assessments. Third, many queries in Internet search engine logs will reflect the limitations of existing search engines that is, web search engines are not designed to connect disparate pieces of information. Users implicitly adapt to this limitation by submitting few queries that reference multiple database entities. Five IMDb queries are outliers because they include an exact quote from a movie. Omitting these queries reduces the maximum number of terms in any query to 7 and the mean number of terms per query to 2.91. The statistics for our queries are similar to those reported for web queries [16] and our independent analysis of query lengths from a commercial search engine log [26], which suggest that the queries are similar to real user queries.

B. Benefits

This system uses two metrics to measure system performance. The first is execution time, which is the time elapsed from issuing the query until the system terminates. Because there are a large number of potential results for each query, systems typically return only the top-k results where k specifies the desired retrieval depth. Our second metric is response time, which this define as the time elapsed from issuing the query until i results have been returned by the system . Because this definition is not well-defined when fewer than k results are retrieved by a system performance should not be measured without also accounting for search effectiveness due to tradeoffs between runtime and the quality of search results. Precision is the ratio of relevant results retrieved to the total number of retrieved results. This metric is important because not every result is actually relevant to the query's underlying information need. If fewer than k results are retrieved by a system, it calculate the precision value at the last result. It also use mean average precision (MAP) to measure retrieval effectiveness at greater retrieval depths.

C. Observational Setup

The search techniques were implemented BANKS, DISCOVER, and DISCOVER-II and obtained implementations of BANKS-II, DPBF, BLINKS, and STAR. We corrected a host of flaws in the specifications of these search techniques and the implementation defects that we discovered. With the exception of DPBF, which is written in C++, all the systems were implemented in C#. The implementation of BANKS adheres to its original description except that 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, which provides the best performance for queries with AND semantics [14].

BLINKS's block index was created using breadth-first partitioning and contains 50 nodes per block. STAR uses the edge weighting scheme proposed for undirected graphs. For our experiments, we executed the C# implementations on a Windows machine with dual core 2.4 GHz AMD Opteron 242 processors and 2 GB of RAM. We compiled each system using C# version 4.0 and ran the implementations with Microsoft Visual studio 2010. Due to its Windows bindings, DPBF could not be run on the same machines as the Java implementations. Instead, DPBF was run on a 2.4 GHz Intel Core 2 quad-core processor with 4 GB of RAM running Windows 7.

It is used Microsoft SQL server 2008 as our database management system. For all the systems, we limit the size of results to 5 nodes (tuples) and impose a maximum execution time of 30 minutes. If the system has not terminated after this time limit, we stop its execution and denote it as a timeout exception. This threshold seems more than adequate for capturing executions that would complete within a reasonable amount of time. If a system exhausts the total amount of virtual memory, we mark it as failing due to excessive memory requirements.

V. Observation

Graph lists the number of queries executed successfully by each system for our datasets and also the number and types of exceptions we encountered. Of interest is the number of queries that either did not complete execution within 1 hour or exhausted the total amount of virtual memory. Most search techniques complete all the MONDIAL queries with mean execution times ranging from less than a second to several hundred seconds. Results for IMDb and Wikipedia are more troubling. Only DISCOVER and DISCOVER-II complete any IMDb queries, and their mean execution time is several minutes. DPBF joins these two systems by completing all the Wikipedia queries, but all three systems' mean execution times are less than ideal, ranging from 6–30 seconds. To summarize these results, existing search techniques provide reasonable performance only on the smallest dataset (MONDIAL). Performance degrades significantly when we consider a dataset with hundreds of thousands of tuples (Wikipedia) and becomes unacceptable for millions of tuples (IMDb). The memory consumption for these algorithms is considerably higher than reported, preventing most search techniques from searching IMDb.

A. Execution Time

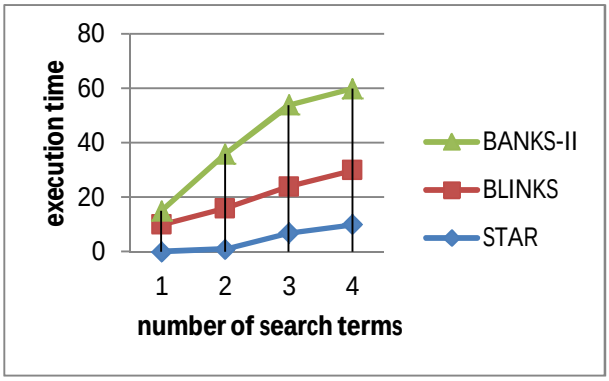

Fig 3. Mean execution time vs number of search terms.

1) Number of search terms: A number of evaluations [8], [14], [15], [17] report mean execution time for queries that contain different numbers of search terms to show that performance remains acceptable even when queries contain more keywords. Note that some systems fail to complete some queries, which accounts for the omissions in the graph. As evidenced by the graph, queries that contain more search terms require more time to execute on average than queries than contain fewer search terms. The relative performance among the different systems is unchanged. These results are similar to those published in previous evaluations. DISCOVER-II to illustrate the range in execution times encountered across the various queries. As evidenced by these graphs, several queries have execution times much higher than the rest. These queries give the system the appearance of unpredictable performance, especially when the query is similar to another one that completes quickly.

2) Frequency collection: In an effort to better understand another factor that is commonly cited as having a performance impact, we consider mean execution time and the frequency of search terms in the database (Figure 6). The results are surprising: execution time appears relatively uncorrelated with the number of tuples containing search terms. This result is counter-intuitive, as one expects the time to increase when more nodes (and all their relationships) must be considered. One possible explanation for this phenomenon is that the search space in the interior of the data graph (i.e., the number of nodes that must be explored when searching) is not correlated with the frequency of the keywords in the database. He et al. [13] imply the opposite; we believe additional experiments are warranted as part of future work.

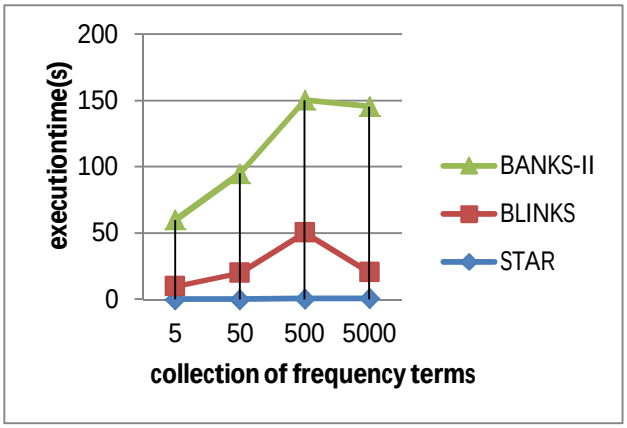

Fig 4. Execution time vs mean frequency

3) Depth of retrieval: Continuing this analysis to higher retrieval depths is not particularly useful

given the small size of the MONDIAL database and given that most systems identify all Continuing this analysis to higher retrieval depths is not particularly useful given the small size of the MONDIAL database and given that most systems identify all.

B. Response Time

In addition to overall search time, the response time of a keyword search system is of critical importance. Systems that support top-k query processing need not enumerate all possible results before outputting some to the user. Outputting a small number of results (e.g., 10) allows the user to examine the initial results and to refine the query if these results are not satisfactory. Interestingly, the response time for most systems is very close to the total execution time, particularly for k = 10. The ratio of response time to the total execution time provided in the table shows that some scoring functions are not good at quickly identifying the best search results. For example, DISCOVER-II identifies the highest ranked search result at the same time as it identifies the tenth ranked result because its bound on the possible score of unseen results falls very rapidly after enumerating more than k results.

In general, the proximity search systems manage to identify results more incrementally than the schema-based approaches. Another issue of interest is the overhead required to retrieve additional search results. Exception of BANKS-II, the total overhead is minimal less than a few seconds. In the case of STAR, the percentage slowdown is high, but this value is not significant given that the execution time is so low.

C. Memory consumption

Limiting the graph-based approaches to 2 GB of virtual memory might unfairly bias our results toward the schema based approaches. The schema-based systems offload much of their work to the underlying database, which swaps temporary data (e.g., the results of a join) to disk as needed. Hence, DISCOVER and DISCOVER-II might also require a significant amount of memory, and a more fair evaluation would allow the graph-based techniques to page data to disk. To investigate this possibility, we ran all the systems with 2 GB of physical memory and 5 GB of virtual memory. Note that once a system consumes the available physical memory, the operating system's virtual memory manager is responsible for paging data to and from disk. The precipitous drop in execution time suggests that Java's garbage collector was responsible for the majority of BLINKS's execution time, and this overhead was responsible for BLINKS's poor performance. The other graph-based systems do not significantly improve from the additional virtual memory. In most cases, we observed severe thrashing, which merely transformed memory exceptions into timeout exceptions.

i. Initial Memory Consumption: To better understand the memory utilization of the systems particularly the overhead of an in-memory data graph, we measured each system's memory footprint immediately prior to executing a query. The schema-based systems consume very little memory, most of which is used to store the database schema. In contrast, the graph-based search techniques require considerably more memory to store their data graph.

VI. Related Work

Existing evaluations of relational keyword search systems are adhoc with little standardization. Webber [33] summarizes existing evaluations with regards to search effectiveness. Although Coffman and Weaver [5] developed the benchmark that we use in this evaluation, their work does not include any performance evaluation. Baid et al. [1] assert that many existing keyword search techniques have unpredictable performance due to unacceptable response times or fail to produce results even after exhausting memory. Our results particularly the large memory footprint of the system confirm this claim. A number of relational keyword search systems have been published beyond those included in our evaluation. Chen et al. [4] and Chaudhuri and Das [3] both presented tutorials on keyword search in databases. Yu et al. [35] provides an excellent overview of relational keyword search techniques. Liu et al. [21] and SPARK [22] both propose modified scoring functions for schema-based

keyword search. SPARK also introduces a skyline sweep algorithm to minimize the total number of database probes during a search. Qin et al. [27] further this efficient query processing by exploring semi-joins. Baid et al. [1] suggest terminating the search after a predetermined period of time and allowing the user to guide further exploration of the search space. In the area of graph-based search techniques, EASE [20] indexes all r-radius Steiner graphs that might form results for a keyword query. Golenberg et al. [12] provide an algorithm that enumerates results in approximate order by height with polynomial delay. Dalvi et al. [6] consider keyword search on graphs that cannot fit within main memory. CSTree [19] provides alternative semantic the compact Steiner tree to answer search queries more efficiently.

In general, the evaluations of these systems do not investigate important issues related to performance (e.g., handling data graphs that do not fit within main memory). Many evaluations are also contradictory, for the reported performance of each system varies greatly between different evaluations. Our experimental results question the validity of many previous evaluations, and we believe our benchmark is more robust and realistic with regards to the retrieval tasks than the workloads used in other evaluations. Furthermore, because our evaluation benchmark is available for other researchers to use, we expect our results to be repeatable.

VII. Conclusion and Future Work

It is not like many of the evaluations reported in the literature, it is designed to investigate not the underlying algorithms but the overall, end-to-end performance of these retrieval systems. Hence, it favor a realistic query workload instead of a larger workload with queries that are unlikely to be representative by randomly selecting Overall, the performance of existing relational keyword search systems is somewhat disappointing, particularly with regard to the number of queries completed successfully in our query workload, Given previously published results it is especially surprised by the number of timeout and memory exceptions that we witnessed. Because it has larger execution times might only reflect choice to use larger datasets, we focus on two concerns that we have related to memory utilization. First, no system admits to having a large memory requirement.

In fact, memory consumption during a search has not been the focus of any previous evaluation. To the best of our knowledge, only two papers [6], [18] have been published in the literature that make allowances for a data graph that does not fit entirely within main memory. Given that most existing evaluations focus on performance, handling large data graphs should be well-studied. Kasneci et al. [18] show that storing the graph on disk can also be extremely expensive for algorithms that touch a large number of nodes and edges. Second, our results seriously question the scalability of these search techniques. MONDIAL is a small dataset that contains fewer than 20K tuples. While its schema is complex, we were not expecting failures due to memory consumption.

Although we executed our experiments on machines that have a small amount of memory by today's standards, scalability remains a significant concern. If 2 GB of memory is not sufficient for MONDIAL, searching our IMDb subset will require ' 200 GB of memory and searching the entire IMDb database would require ' 5 TB. Without additional research into high-performance algorithms that maintain a small memory footprint, these systems will be unable to search even moderately-sized databases and will never to suitable for large databases like social networks or medical health records. 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 , which underscores the need for a progression of datasets that will allow researchers to make progress toward this objective. Creating a subset of the original dataset is common, but we are not aware of any work that identifies how to determine if a subset is representative of the original dataset. 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. It must design algorithms, data structures, and implementations that recognize that storing a complete graph representation of a databases.

It can also use for fully structured or fully semi-structured databases. It will produce or achieve another milestone in evaluation of the systems standards of the keyword search process of the scheme.

VIII.References

1. Baid, I. Rae, J. Li, A. Doan, and J. Naughton, "Toward Scalable Keyword Search over Relational Data," Proceedings of the VLDB Endowment, vol. 3, no. 1, pp. 140–149, 2010.
2. G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan,"Keyword Searching and Browsing in Databases using BANKS," in Proceedings of the 18th International Conference on Data Engineering, ser. ICDE '02, February 2002, pp. 431–440.
3. S. Chaudhuri and G. Das, "Keyword Querying and Ranking in Databases," Proceedings of the VLDB Endowment, vol. 2, pp. 1658–1659, August 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1687553.
4. Y. Chen, W. Wang, Z. Liu, and X. Lin, "Keyword Search on Structured and Semi-Structured Data," in Proceedings of the 35th SIGMOD International Conference on Management of Data, ser. SIGMOD '09, June 2009, pp. 1005–1010.
5. J. Coffman and A. C. Weaver, "A Framework for Evaluating Database Keyword Search Strategies," in Proceedings of the 19th ACM International Conference on Information and Knowledge Management, ser. CIKM '10, October 2010, pp. 729–738. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871531
6. B. B. Dalvi, M. Kshirsagar, and S. Sudarshan, "Keyword Search on External Memory Data Graphs," Proceedings of the VLDB Endowment, vol. 1, no. 1, pp. 1189–1204, 2008.
7. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," Numerische Mathematik, vol. 1, no. 1, pp. 269–271, 1959.
8. B. Ding, J. X. Yu, S. Wang, L. Qin, X. Zhang, and X. Lin, "Finding Topk Min-Cost Connected Trees in Databases," in ICDE '07: Proceedings
9. of the 23rd International Conference on Data Engineering, April 2007, pp. 836–845.
10. S. E. Dreyfus and R. A. Wagner, "The Steiner Problem in Graphs," Networks, vol. 1, no. 3, pp. 195–207, 1971. [Online]. Available: http://dx.doi.org/10.1002/net.3230010302
11. D. Fallows, "Search Engine Use," Pew Internet and American Life Project, Tech. Rep., August 2008, http://www.pewinternet.org/Reports/
12. 2008/Search-Engine-Use.aspx.
13. [11] "Global Search Market Grows 46 Percent in 2009," http: //www.comscore.com/Press Events/Press Releases/2010/1/ Global Search Market Grows 46 Percent in 2009, January 2010.
14. K. Golenberg, B. Kimelfeld, and Y. Sagiv, "Keyword Proximity Search in Complex Data Graphs," in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '08, June 2008, pp. 927–940.
15. H. He, H. Wang, J. Yang, and P. S. Yu, "BLINKS: Ranked Keyword Searches on Graphs," in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD '07, June 2007, pp. 305–316.
16. Hristidis, L. Gravano, and Y. Papakonstantinou, "Efficient IR-style Keyword Search over Relational Databases," in Proceedings of the 29th International Conference on Very Large Data Bases, ser. VLDB '03, September 2003, pp. 850–861.

Hristidis and Y. Papakonstantinou, "DISCOVER: Keyword Search in Relational Databases," in Proceedings of the 29th International

Conference on Very Large Data Bases, ser. VLDB ’02. VLDB Endowment, August 2002, pp. 670–681.

17. B. J. Jansen and A. Spink, “How are we searching the World Wide Web? A comparison of nine search engine transaction logs,” Information Processing and Management, vol. 42, no. 1, pp. 248–263, 2006.
18. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar, “Bidirectional Expansion for Keyword Search on Graph Databases,” In Proceedings OF The 31st International Conference on Very Large Data Bases, Ser. VLDB ’05, August 2005, pp. 505–516.
19. Kasneci, M. Ramanath, M. Sozio, F. M. Suchanek, and G. Weikum, “STAR: Steiner-Tree Approximation in Relationship Graphs,” in Proceedings of the 25th International Conference on Data Engineering, ser. ICDE ’09, March 2009, pp. 868–879.
20. Li, J. Feng, X. Zhou, and J. Wang, “Providing built-in keyword search capabilities in RDBMS,” The VLDB Journal, vol. 20, pp. 1–19, February 2011. [Online]. Available: http://dx.doi.org/10.1007/s00778-010-0188-4
21. [20] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou, “EASE: An Effective 3-in-1 Keyword Search Method for Unstructured, Semi-structured and Structured Data,” in Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’08, June 2008, pp. 903–914.
22. F. Liu, C. Yu, W. Meng, and A. Chowdhury, “Effective Keyword Search in Relational Databases,” in Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’06, June 2006, pp. 563–574.
23. Y. Luo, X. Lin, W. Wang, and X. Zhou, “SPARK: Top-k Keyword Query in Relational Databases,” in Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, ser. SIGMOD ’07, June 2007, pp. 115–126.
24. C. D. Manning, P. Raghavan, and H. Sch¨utze, Introduction to Information Retrieval. New York, NY: Cambridge University Press, 2008.
25. W. May, “Information Extraction and Integration with FLORID: The MONDIAL Case Study,” Universit¨at Freiburg, Institut f¨ur Informatik, Tech. Rep. 131, 1999, available from http://dbis.informatik.uni-goettingen. de/Mondial.
26. Nandi and H. V. Jagadish, “Qunits: queried units for database search,” in CIDR ’09: Proceedings of the 4th Biennial Conference on Innovative Data Systems Research, January 2009.
27. Pass, A. Chowdhury, and C. Torgeson, “A Picture of Search,” in InfoScale ’06: Proceedings of the 1st International Conference on Scalable Information Systems, May 2006.