1 Introduction

Searching for information is an indispensable component of our lives. Web search engines are widely used for searching textual documents, images, and videos. There are also vast collections of structured and semi-structured data both on the Web and in enterprises, such as relational databases, XML, data extracted from text documents, workflows, etc. A natural question to ask is whether we can empower users to effectively access structured data using keyword queries. Ideally the result of a keyword search over structured data will automatically assemble relevant pieces of data that are in different locations but are inter-connected and collectively relevant to the query.

Due to substantial benefits of supporting keyword search on structured data, it becomes an emerging hot area in database research and development. Researchers from different disciplines (e.g., information retrieval and theoretical computer science) are joining the workforce to tackle various challenges in supporting keyword search on structured data.

Semi-supervised clustering aims to improve clustering piece with the help of user-provided side information. One of the most studied types of side information is pair wise constraints, which include must link and cannot-link constraints specifying that two points must or must not belong to the same cluster. A number of previous studies have demonstrated that, in general, such constraints can lead to improved clustering performance. However, if the constraints are selected improperly, they may also degrade the clustering performance. Moreover, obtaining pair wise constraints typically requires a user to manually inspect the data points in question, which can be time consuming and costly. For example, for document clustering, obtaining a must-link or cannot-link constraint requires a user to potentially scan through the documents in question and determine their relationship, which is feasible but costly in time. For those reasons, we would like to optimize the selection of the constraints for semi-supervised clustering, which is the topic of active learning.

While active learning has been extensively studied in supervised learning the research on active learning of constraints for semi-supervised clustering is relatively limited. Most of the existing work on this topic has focused on selecting an initial set of constraints prior to performing semi-supervised clustering. This is not suitable if we wish to iteratively improve the clustering model by actively querying the user. In this paper, we consider active learning of constraints in an iterative framework. Specifically, in each iteration, we determine what is the most important information toward improving the current clustering model and form queries accordingly.

II.RELATED WORK

The implicit assumptions of keyword search that is, the search terms are related complicates the search process because typically there are many possible relationships between search terms. It is frequently possible to include another occurrence of a search term by adding tuples to an existing result. This realization leads to tension between the...
compactness (and consequently performance) and coverage of search results [2].

The main keyword search and relational database layers are surrounded by a Keyword Identification layer, a Results Filtering and Ranking layer, and a Presentation layer. Indeed, an ideal search system has to consider the search task a user desires to conduct, to perform user queries knowing that they may not exactly correspond to the real user information need, to disambiguate search terms, to rank the results [4] of the search process on the basis of relevance for the user, and to visualize these results in the most proper way for the considered search task.

An efficient and scalable keyword search utility for relational databases, is described. DBXplorer returns all rows either from single tables, or by joining tables connected by foreign-key joins such that the each row includes all keywords. It also provides an indexing structure, symbol table which find out the exact position of query keywords in the relational entities. Query answers are constructed through breadth-first enumeration and then join each join tree is mapped. The join trees are ranked by the number of joins.

Proximity is an appropriate measure for the relevance of different data objects when a specific query is not provided. When pair of keyword queries is as a input, Find set and Near set, are expressed, the system finds objects specified in the Find set that have the shortest path connecting them to objects specified in the Near set. The purpose of this system is of textual proximity in IR is applied to an arbitrary database which measures some pairs of objects are closely related. The objects are related by a distance function and the score of an object is computed using its proximity to all objects [7]. For efficient distance computation, all pair’s distances are recomputed in a k-neighbors distance lookup table.

III. Proposed Work

Our previous work did not consider the runtime performance of these search techniques, which is our focus. 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. We conduct an independent, empirical evaluation of the runtime performance of seven relational keyword search techniques. Our results do not substantiate previous claims regarding the scalability and performance of relational keyword search techniques. Our work is the first to combine performance and search effectiveness in the evaluation of such a large number of search techniques. Our experiments do not compare against traditional IR systems (e.g., Apache Lucene) because more traditional systems do not consider the relationships among database tuples, which is an important aspect of relational keyword search.

![Fig. 1 Main Diagram Of Proposed Technique](image-url)

A. Repository Creation

Repository commonly refers to a position for storage, often for safety or conservation. In information technology, a repository is a central place in which an aggregation of data is kept and maintained in an organized way, usually in computer storage. The term is from the Latin repositories, a vessel or cavity in which things can be placed, and it can mean a place where things are collected. Depending on how the term is used, a repository may be directly reachable to users or may be a place from which specific databases, files, or documents are obtained for further rearrangement or allocation in a network. A storehouse may be just the aggregation of data itself into some available place of storage or it may also imply some ability to selectively extract data. In our paper the admin has rights to create repository. Admin can store the information what user wants to search.

B. Hierarchical Navigation Model

Many solutions have been proposed to address query result problems. These approaches can be broadly classified into two classes: ranking and categorization which can also be combined. Ranking presents the user with a list of results ordered by some metric of relevance or by content similarity to a result or a set of results. In categorization query results are grouped based on hierarchies, keywords, tags, or...
attribute values. User studies have demonstrated the usefulness of categorization in finding relevant results of exploratory queries. While ranked results are useful when the ranking function is aligned with user preferences or the result list is small in size, categorization is generally employed by users when ranking fails or the query is too “broad”.

C. Best Edge Cut

Optimal Algorithm for Best Edge Cut the Opt-Edge Cut algorithm to compute the minimum expected navigation cost (and the Edge Cut that achieves it) traverses the navigation tree in post order and computes the navigation cost bottom-up starting from the leaves. For each node n, the algorithm enumerates and stores the list of all possible Edge Cuts for the sub tree rooted at n, and the list of all possible sets that node n can be annotated with.

D. Search Module

This module gives the result of which user looking to search. It gives the detail information so that user given the query. Since a query may give lakhs of output. So here we displaying that the number of output is coming in tree view so that user can easily find out the exact result.

IV. Results And Discussion

Supporting keyword search is not only helpful for users to access a single database, but also benefits information integration. As the number of potentially-related data sources continues to grow rapidly, the existing approach of using pre-defined forms and associated query templates cannot adequately support diverse data sources and meet diverse user needs. Keyword search provides a light-weight mechanism to access multiple data sources without labor intensive information integration upfront.

A. Database Selection

We will discuss techniques that summarize underlying databases by a keyword relationship graph, and select the most relevant data sources with respect to a user keyword search based on derived summaries.

B. Query Generation

We will discuss techniques that allow a casual user to author new query templates and Web forms by posing keyword searches. The keyword searches are matched against source relations and their attributes to create multiple ranked queries linking the keyword matches. The set of queries is attached to a Web query form, which can be reused by anyone with related information needs.

C. Analytical Processing

Database Selection (D) < Query Generation (Q)

Online Analytical Processing tools provide elaborate query languages that allow users to group and aggregate data in various ways, and to explore interesting trends and patterns in the data. However, the complexity of issuing such analytic queries is overwhelming. It is highly desirable, yet very challenging, to combine intuitive keyword-based search with the power of OLAP, to allow users to easily analyze complex data.

In our proposed work, to calculate query generation value (Q) and Database Selection value (D) then also find the ratio of final value for measuring and maintain the effective outputs.

Efficiency (e) =\frac{\text{Database Selection (D)} - \text{Query Generation (Q)}}{2*100}

![Fig 3 Comparison of structured and semi structured data](image)

V. 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 improvements claimed by previous evaluations. These conclusions are consistent with previous evaluations that demonstrate the poor runtime performance of existing search techniques as a prelude to a newly-proposed approach.

References