RESEARCH ARTICLE OPEN ACCESS

# Avoidance of Ranking Capabilities in Retrieval of Queries on Hidden-Web Text Databases

S K.Rubeena[1], T. Srinivasa Rao[2]
Pursuing M.Tech(cse), Department of Computer Science ,VVIT Nambur (V), Guntur (Dt.).
Associate. Professor, Department of Computer Science, VVIT Nambur (V), Guntur (Dt.).

**ABSTRACT**
Many online or local data sources provide powerful querying mechanisms but limited ranking capabilities. For instance, Pub Med allows users to submit highly expressive Boolean keyword queries, but ranks the query results by date only. However, a user would typically prefer a ranking by relevance, measured by an information retrieval (IR) ranking function. A naive approach would be to submit a disjunctive query with all query keywords, retrieve all the returned matching documents, and then re-rank them. Unfortunately, such an operation would be very expensive due to the large number of results returned by disjunctive queries. In this paper, we present algorithms that return the top results for a query, ranked according to an IR-style ranking function, while operating on top of a source with a Boolean query interface with no ranking capabilities (or a ranking capability of no interest to the end user). The algorithms generate a series of conjunctive queries that return only documents that are candidates for being highly ranked according to relevance metric. Our approach can also be applied to other settings where the ranking is monotonic on a set of factors (query keywords in IR) and the source query interface is a Boolean expression of these factors. Our comprehensive experimental evaluation on the Pub Med database and a TREC data set show that we achieve order of magnitude improvement compared to the current baseline approaches.
**Index Terms:** Hidden-web databases, keyword search, top-k ranking

## I. INTRODUCTION

MANY online or local data sources provide powerful querying mechanisms but limited ranking capabilities. For instance, PubMed1 allows users to submit Boolean keyword queries on the biomedical publications database, but ranks the query results by publication date only.

Similarly, the US Patent and Trademark Office (USPTO)2 allows Boolean keyword queries or searching patents but only ranks by patent date. Furthermore, job search databases, such as the job search of LinkedIn,3 allow users to sort job listings by date or title (alphabetically), but not by IR relevance of the job posting to the submitted query. As a more recent example, the micro-blogging service Twitter4 offers a highly expressive Boolean search interface but ranks the results by date only. In most cases, these sources do not allow downloading and indexing of data or the size of the underlying database makes any comprehensive download an expensive operation. Often, the user prefers a ranking other than the default sorting (e.g., by date) provided by the source. For instance, a user of the PubMed or USPTO Websites may prefer a ranking by relevance, measured by an Information Retrieval (IR) ranking function, as opposed to a date-based retrieval. Given that traditional IR ranking functions like Ok ap and BM25 implicitly assume disjunctive (OR) semantics, the naive approach would be to submit to the database a disjunctive query with all query keywords, retrieve all the returned documents, and then rank them according to the relevance metric of choice. However, this would be very expensive due to the large number of results returned by disjunctive queries. For example, consider the query “immunodeficiency virus structure,” an example query used to teach information specialists how to search the PubMed database. Executing the corresponding disjunctive query “immunodeficiency OR virus OR structure” on PubMed returns 1,451,446 publication results. Downloading and ranking them is infeasible for an interactive query system, even if the source is on the local network. The problem becomes even more critical if we use the public web services provided by PubMed for programmatic (API) access over the web. Given the large overhead incurred when retrieving publications, PubMed imposes quotas on the amount of data an application can retrieve per minute, rendering infeasible any attempt to download large number of documents. To overcome such problems, in this paper, we present algorithms to compute the top results for an IR ranked query, over a source with a Boolean query interface but without any ranking capabilities (or with a ranking function that is generally uncorrelated to the user’s ranking: e.g., by date). A key idea behind our technique is to use a probabilistic modeling approach, and estimate the distribution of document scores that are expected to be returned by the database.

**Problem Definition**

We want to devise a scheme for retrieving from D the top-k documents, ranked according to. The trivial solution is to send an extremely broad disjunctive query, returning all documents that have a nonzero score. Then, we can retrieve the documents, examine their contents, and re-rank them locally before presenting the results to the user. Unfortunately, this is a very time-consuming solution. Therefore, our objective is to construct a query sequence q1; q2; . . . ; qv of Boolean queries, that can be submitted to the database, retrieve as few documents as possible, and still contain all the documents that would be in the top-k results.

**Literature Survey**

The user prefers a ranking other than the default sorting (e.g., by date) provided by the source. For instance, a user of the Pub Med or USPTO Websites may prefer a ranking by relevance, measured by an Information Retrieval (IR) ranking function, as opposed to a date-based retrieval. Given that traditional IR ranking functions like Okapi and BM25 implicitly assume disjunctive (OR) semantics, the naive approach would be to submit to the database a disjunctive query with all query keywords, retrieve all the returned documents, and then rank them according to the relevance metric of choice. However, this would be very expensive due to the large number of results returned by disjunctive queries. For example, consider the query “immunodeficiency virus structure,” an example query used to teach information specialists how to search the Pub Med database. Executing the corresponding disjunctive query “immunodeficiency OR virus OR structure” on Pub Med returns 1,451,446 publication results. Downloading and ranking them is infeasible for an interactive query system, even if the source is on the local network. The problem becomes even more critical if we use the public web services provided by Pub Med for programmatic (API) access over the web. Given the large overhead incurred when retrieving publications, Pub Med imposes quotas on the amount of data an application can retrieve per minute, rendering infeasible any attempt to download large number of documents.

**Disadvantages:**

The problem becomes even more critical if we use the public web services provided by Pub Med for programmatic (API) access over the web. Given the large overhead incurred when retrieving publications, Pub Med imposes quotas on the amount of data an application can retrieve per minute, rendering infeasible any attempt to download large number of documents.

**Proposed System**

To overcome such problems, in this paper, we present algorithms to compute the top results for an IR ranked query, over a source with a Boolean query interface but without any ranking capabilities (or with a ranking function that is generally uncorrelated to the user’s ranking: e.g., by date). A key idea behind our technique is to use a probabilistic modeling approach, and estimate the distribution of document scores that are expected to be returned by the database. Hence, we can estimate what are the minimum cutoff scores for including a document in the list of highly ranked documents. To achieve this result over a database that allows only query-based access of documents, we generate a querying strategy that submits a minimal sequence of conjunctive queries to the source. (Note that conjunctive queries are cheaper since they return significantly fewer results than disjunctive ones.) After every submitted conjunctive query we update the estimated probability distributions of the query keywords in the database and decide whether the algorithm should terminate given the user’s results confidence requirement or whether further querying is necessary; in the latter case, our algorithm also decides which is the best query to submit next. For instance, for the above query “immunodeficiency virus structure,” the algorithm may first execute “immunodeficiency AND virus AND structure,” then “immunodeficiency AND structure” and then terminate, after estimating that the returned documents contain all the documents that would be highly ranked under an IR-style ranking mechanism. As we will see, our work fits into the “exploration versus exploitation” paradigm, since we iteratively explore the source by submitting conjunctive queries to learn the probability distributions of the keywords, and at the same time we exploit the returned “document samples” to retrieve results for the user query.

**Advantages:**

1. We define the novel problem of applying ranking on top of sources with no ranking capabilities by exploiting their query interface.
2. We describe sampling strategies for estimating the relevance of the documents retrieved by different keyword queries. We present a static sampling approach and a dynamic sampling approach that simultaneously executes the query, estimates the parameters required for efficient query execution, and compensates for the biases in the sampling process.
3. We present algorithms that, given a user confidence input, retrieve a minimal number of results from the source through submitting high-selectivity (conjunctive) queries, so that the user’s confidence requirement is satisfied.
4. We experimentally evaluate our algorithms using the Pub Med database and examine two settings: 1) the remote setting, where we use web services

to query the database, and 2) the local setting where we query a locally installed subset of Pub Med. Our results show an order of magnitude improvement compared to the naive query evaluation approach.

## II. SYSTEM ANALYSIS

A feasibility study is a high-level capsule version of the entire System analysis and Design Process. The study begins by classifying the problem definition. Feasibility is to determine if it's worth doing. Once an acceptance problem definition has been generated, the analyst develops a logical model of the system. A search for alternatives is analyzed carefully. There are 3 parts in feasibility study.

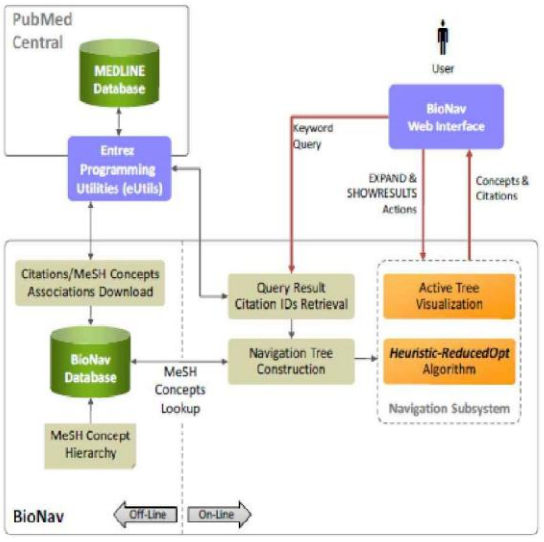
### Technical Feasibility

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis .Understand the different technologies involved in the proposed system before commencing the project that have to be very clear about what are the technologies that are to be required for the development of the new system. Find out whether the organization currently possesses the required technologies. Proposed project is beneficial only if it can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance. Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems .Have the user been involved in the planning and development of the project? Early involvement reduces the chances of resistance to the system and in general and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered. In the existing manual system, the new system was considered to be operational feasible. Economic feasibility attempts to weigh the costs of developing and implementing a new system. This feasibility study gives the top management the economic justification for the new system. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be a useful point of reference to compare actual costs as the project progresses. There could be various types of intangible benefits on account of automation. .

### System Architecture


PubMed Central
MEDLINE Database
Entrez Programming Utilities (eUtils)
User
BioNav Web Interface
Keyword Query
EXPAND & SHOWRESULTS Actions
Concepts & Citations
Citations/MeSH Concepts Associations Download
Query Result Citation IDs Retrieval
Active Tree Visualization
BioNav Database
MeSH Concepts Lookup
Navigation Tree Construction
Heuristic-ReducedOpt Algorithm
Navigation Subsystem
MeSH Concept Hierarchy
BioNav
Off-Line
On-Line
Fig1.BioNav System Architecture

### Modules

There are 2 modules 1. Query Model,2. Data Source Model.

### Query Model

Consider a text database D with documents d; . . . ; dm. The user submits a keyword query Q ¼ ft1 . . . tng containing the terms t1 . . . tn. The answer to the query is a list of the top k documents; the documents are ranked according to a relevance score, which estimates the relevance of a document d to the query Q. The score of a document can be computed using any of the well studied tf.idf scoring functions like BM25 and Okapi . The key arguments of a tf.idf function are the term frequency (tf), the document frequency (df) and the document length (dl). The term frequency; is the number of times that the word t appears in document d. The document frequency; DÞ is the number of documents in D that contain t. the tf.idf ranking function is score the size of the database D. In our experiments, we use the Okapi scoring function, although any other tf.idf function could be used. For simplicity though we use the basic tf.idf scoring function as the running example.

### Data Source Model

We assume that database D is only accessible through a Boolean query interface and we do not have direct access to the underlying documents. The query interface evaluates the Boolean query Q and returns the documents ranked using a non desirable ranking function, e.g., by date (as is the case for Pub Med and USPTO). For instance, if the user query is Q ¼ [anemia, diabetes, sclerosis], then we can submit to the data source queries [anemia AND diabetes AND sclerosis], q2 ¼ [anemia AND diabetes AND NOT sclerosis diabetes OR sclerosis], and so on. The returned results are guaranteed to match the Boolean conditions but the documents are not expected to be ranked in any useful manner.

## III. SYSTEM DESIGN

Design is a meaningful engineering representation of something that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where quality is fostered in software engineering. Design is the perfect way to accurately translate a customer's requirement in to a finished software product. Design creates a representation or model, provides detail about software data structure, architecture, interfaces and components that are necessary to implement a system.Design is multi-step process that focuses on data structure software architecture, procedural details and interface between modules. The design process also translates the requirements into the presentation of software that can be accessed for quality before coding begins. Computer software design changes continuously as new methods; better analysis and broader understanding evolved. Software Design is at relatively early stage in its revolution.

Therefore, Software Design methodology lacks the depth, flexibility and quantitative nature that are normally associated with more classical engineering disciplines. However techniques for software designs do exist, criteria for design qualities are available and design notation can be applied.

**UML Diagrams**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules. The Unified Modeling Language (UML) is a standard visual modeling language intended to be used for modeling business and similar processes, analysis, design, and implementation of software-based systems.UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

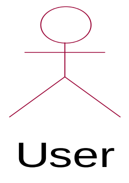
**Use Case Diagram**

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior.
A use-case diagram can contain:

- Actors ("things" outside the system)
- Use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.
  Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented.

**Graphical Depiction:**

An actor is a stereotype of a class and is depicted as a "stickman" on a use-case diagram.



## IV. IMPLEMENTATION

A programming tool or software tool is a program or application that software developers use to create, debug, maintain, or otherwise support other programs and applications. The term usually refers to relatively simple programs that can be combined together to accomplish a task. The Chapter describes about the software tool that is used in our project**.**

## V. TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets the requirements and user expectations and does not fail in an unacceptable manner. There are various types of testing. Each test type addresses a specific testing requirement.

**TEST CASES:**
**Test Case 1:**
Input: Without giving any URL.
Output: It will display an exception.
**+VE TEST CASES**

| **S.No** | **Test case Description** | **Actual value** | **Expected value** | **Result** |
|---|---|---|---|---|
| **1** | Create the new user registration process | New user created successfully | To update the database in oracle | True |

| | | | | |
|---|---|---|---|---|
| **2** | Enter the login information | Enter the username and password | Gets the home page | True |
| **3** | Enter the Disjunctive query | It can extract the results based on Logical OR operation | Huge documents are displayed here | True |
| **4** | Enter the Conjunctive query | Perform the query operation that is called Logical | It can extracts minimized results | True |

**Test Case 2:**
Input: Templates are not selected.
Output: Web template training complete with 0 files.
**-VE TEST CASES**

| **S.No** | **Test case Description** | **Actual value** | **Expected value** | **Result** |
|---|---|---|---|---|
| **1** | Create the new user registration process | New user is not created successfully | Data Values is not update in oracle. | False |
| **2** | Enter the login information | Enter the username and password | Error page is displayed here. | False |
| **3** | Enter the Disjunctive query | It cannot extract the results based on Logical OR operation | Huge documents are not displayed here | False |
| **4** | Enter the Conjunctive query | Perform the query operation that is called Logical AND | It is not extracts minimized results | False |

## VI. CONCLUSION

We presented a framework and efficient algorithms to build a ranking wrapper on top of a documents data source that only serves Boolean keyword queries. Our algorithm submits a minimal sequence of conjunctive queries instead of a very expensive disjunctive one. Our comprehensive experimental evaluation on the Pub Med database shows that we achieve order of magnitude improvement compared to the baseline approach.In our work, we are trying to maximize the payoff/exploitation of each query (which is the number of new, relevant top-k documents that the query retrieves) while minimizing the expense/exploration (number of queries sent, and documents retrieved).

## REFERENCES

[1] RamezElmasri, ShamkantB. Navathe, ”Fundamental of Database Systems”, Pearson Education.
[2] A. Ntoulas, P. Zerfos, and J. Cho, “Downloading Textual Hidden Web Content by Keyword Queries,” Proc. Fifth ACM and IEEE Joint Conf. Digital Libraries (JCDL ’05), 2005.
[3] Database Management Systems,Peter Rob, Carlos Coronel,Cengage Learning.
[4] Z. Lu, W. Kim, and W.J. Wilbur, “Evaluating Relevance Ranking Strategies for Medline Retrieval,” J. Am. Medical Informatics Assoc., vol. 16, no. 1, pp. 32-36, 2009.
[5] Silber Schatz.Korth,Database System Concepts, Tata Mc Graw Hill.
[6] A. Singhal, “Modern Information Retrieval: A Brief Overview,” Bull. IEEE CS Technical Committee on Data Eng., vol. 24, no. 4, pp. 35-42, http://singhal.info/ieee2001.pdf, 2001.
[7] Principles of Database Systems J.D.Ullman,Galotia Pub.1994.