# On Scalable Information Retrieval Systems

*(Invited Paper)*

Ophir Frieder & David Grossman
*Illinois Institute of Technology*
*{frieder, grossman}@iit.edu*

Abdur Chowdhury
*America Online*
*cabdur@aol.com*

## Abstract

*Scalable information retrieval systems are crucial to meeting the growing volumes of data. We describe work done to facilitate scalability by reducing duplication, providing integration with structured data, and supporting integration and question answering via an intranet mediator. All examples given are taken directly from prior and on-going efforts in the IIT Information Retrieval Laboratory in collaboration with AOL and NCR.*

## 1. Introduction

The development of *scalable* computerized *information retrieval* systems is of concern since the advent of computing and information processing. In the computer science discipline, we are accustomed to gradually varying definitions particularly as they pertain to size and time. What was considered large ten years ago is now considered constrained. For example, a personal computer configured with a gigabyte of RAM today is minimally configured whereas ten years ago would be considered a dream. Thus, when addressing the topic of *scalable information retrieval* systems, one must bind the terms to current day dimensions.

We begin by defining our scope. Information retrieval is the identification of documents relevant to a user's query. Scalable information retrieval systems are those systems that integrate significant volumes of data whose sources of data generation are geographically distributed. For us, scalability in terms of data volume is on the order of the text data volume available on the World Wide Web, currently approximated on the order of tens of terabytes, and query processing rates of thousands of queries per second.

AOL web search, for example, uses up to 50 different sources for each query on their web service when dealing with various queries. Additionally, query volumes range from several queries per second to thousands of queries per second for high volume locations. Thus, two requirements must be made for retrieval systems, first they must be scalable, and second infrastructure must be provided to tie many different collections of data together seamlessly.

In the remainder of this paper, we describe some of the research problems encountered in developing a scalable search service. In our context, a scalable search service is a collection of data sources, in which a single user interface is provided and multiple data sources are used to bring users closer to the information needed in a manner that best helps their information need. First, we examine the duplicate data problem that search services encounter as they integrate multiple data sources into a single content source. Next, we examine the integration of structured and unstructured data into a single integrated search solution. Lastly, we examine the combination of multiple sources of data (intranet mediator) producing a single search service. All examples given are taken directly from prior and on-going efforts in the IIT Information Retrieval, some of which are in collaboration with AOL and NCR.

## 2. Duplicate Document Detection

The data, simply by the nature of how and where they are generated, are distributed. Furthermore, key references (documents) are replicated across multiple sites. For a general overview of the nature of distributed information systems from a data engineering perspective, please see [1]. In information retrieval systems, replication of data poses several difficulties. For starters, when users post queries to the system, they wish to get unique answers; ten near-duplicate copies of the same material is not meaningful to the user.

Duplicates also affect system performance. Document relevance to a query is computed based on the uniqueness of the terms comprising it, namely as a function of the *inverse document frequency (idf)* of its terms. The idf value of a term is computed as a function of the inverse of the number of times a term appears in a document collection. The fewer the documents containing the term, the higher is its associated idf value. (For further details on idf and other information retrieval issues, please see [2].) Thus, if multiple copies of a given document were to be maintained, terms that should be considered unique, namely having a high idf

value, would not be evaluated as unique since they appear in multiple documents. Finally, retrieving multiple copies obviously increases the I/O, introducing longer processing times. For all of these reasons, duplicate elimination is necessary.

Our approach to duplicate elimination, originally described in [3], is as follows. We read a document as input, parse it, retain one copy of each term comprising the document, remove some of the terms according to a variation of idf threshold techniques, sort the terms in Unicode ordering, and ultimately, create a single hash value for the remaining list of terms representing the document. All documents resulting in the same hash value are deemed as duplicates.

The identification of duplicates is handled through inserts into a tree or hash table. Any collisions of hash values represent duplicates, and the document identifiers are stored in that node of the tree or hash bucket. A scan through the tree or hash table produces a list of all clusters of duplicates, where a node contains more than one document.

The overall runtime of our approach is *(O(d log d))* in the worst case where all documents *(d)* are duplicates of each other and *(O(d))* otherwise. We favorably compared against all known duplicate detection algorithms in terms of efficiency and accuracy. In all experiments to date, this approach was more efficient in terms of processing times and more accurate in terms of determining duplicates and near-duplicates. This approach is currently in commercial deployment at a leading content provider.

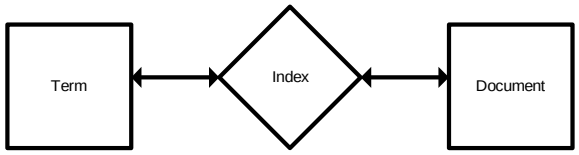


**Figure 1: Entity Diagram**

```
<DOC>
<DOCNO>          0028 </DOCNO>
<HEAD>Stocks Up In Tokyo</HEAD>
<DATELINE>TOKYO (AP) </DATELINE>
<TEXT>
The Nikkei Stock Average closed at 29,754.73
 up 156.92 points on the Tokyo Stock Exchange
 on Wednesday.
</TEXT>
</DOC>
```

**Figure 2: Sample TREC Document**

## 3. Integration

To reduce I/O thereby improve search speeds, all search engines have at their core an *inverted index*, a highly efficient read-only data structure that maps a given term to a list of documents that contain the term. We first describe our efforts that integrate data by

**DOCUMENT**

| docID | docname | headline | dateline |
|---|---|---|---|
| 28 | AP881214-0028 | Stocks Up In Tokyo | TOKYO (AP) |

**INDEX**

| docID | termcnt | term |
|---|---|---|
| 28 | 1 | nikkei |
| 28 | 2 | stock |
| 28 | 1 | average |
| 28 | 1 | closed |
| 28 | 2 | points |
| 28 | 1 | up |
| 28 | 1 | tokyo |
| 28 | 1 | exchange |
| 28 | 1 | wednesday |

**TERM**

| term | df | idf |
|---|---|---|
| average | 2265 | 1.08 |
| closed | 2208 | 1.08 |
| exchange | 2790 | 1.00 |
| nikkei | 234 | 2.07 |
| points | 1627 | 1.23 |
| stock | 2674 | 1.00 |
| tokyo | 725 | 1.58 |
| up | 12746 | 0.30 |
| Wednesday | 6417 | 0.60 |

**Figure 3: Relations that model the TREC document**

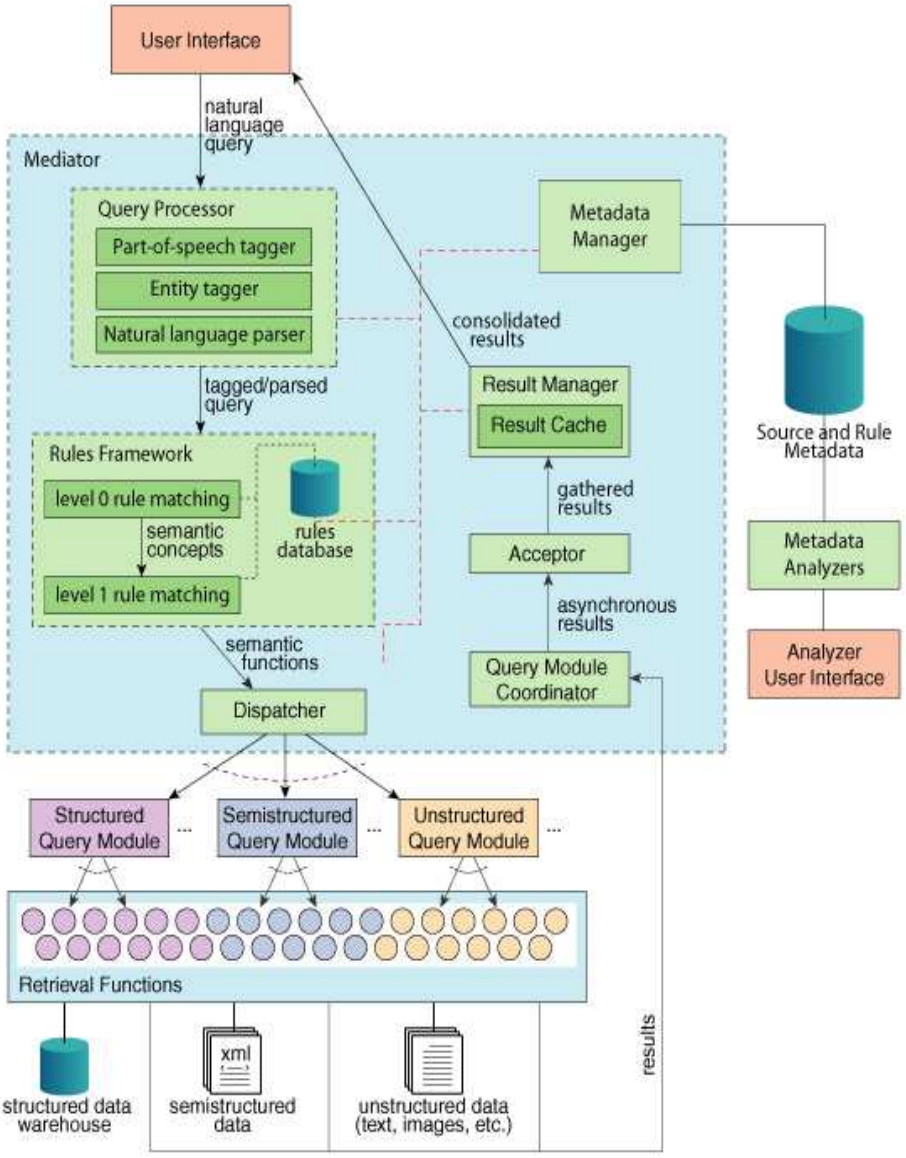


**Figure 4: Architecture**

modeling an inverted index as a relation. This essentially treats data integration as a pure application of the relational model. Since relational systems are designed for structured data, mapping the inverted index on to relations yields an integrated text and structured data search engine. Subsequently, we describe recent work on the development of an *intranet mediator* [4] that is able to integrate data by leveraging existing work on schema reconciliation.

### 3.1. Integrating Data with the Relational Model

Given the need to integrate structured data and text in a scalable fashion, we developed a system named SIRE (Scalable Information Retrieval Engine) [5, 6]. SIRE is the mapping of an inverted index structure on to a set of relations with a corresponding set of relational scripts. An entity-relationship diagram (ERD) representing this mapping of terms and documents is shown in Figure 1. A given term can appear in many documents, and one document naturally has many terms. Hence, there is a many-many relationship between terms and documents.

Each object in the ERD corresponds to a relation from the following three relations:

DOCUMENT (*docID, docname, headline, dateline)*
INDEX(docID, *termcnt, term*)
TERM(*term, df, idf*)

In Figure 2, we illustrate a sample document, and in Figure 3, we illustrate how this document can be mapped onto the three relations. The DOCUMENT relation contains metadata about the document such as the dateline, headline, and document name. The TERM relation has a tuple for each distinct term in the collection with its corresponding term weights: document frequency and inverse document frequency. These weights are used by many information retrieval systems. Other additional weights are easily incorporated into this model.

The index relation models the inverted index. It contains one entry for each element that would ordinarily be stored in a posting list of the inverted index. The posting list typically is a list of entries that indicate which documents contain a given term, and how often the term appears in the document. Here, a tuple is stored in the INDEX relation.

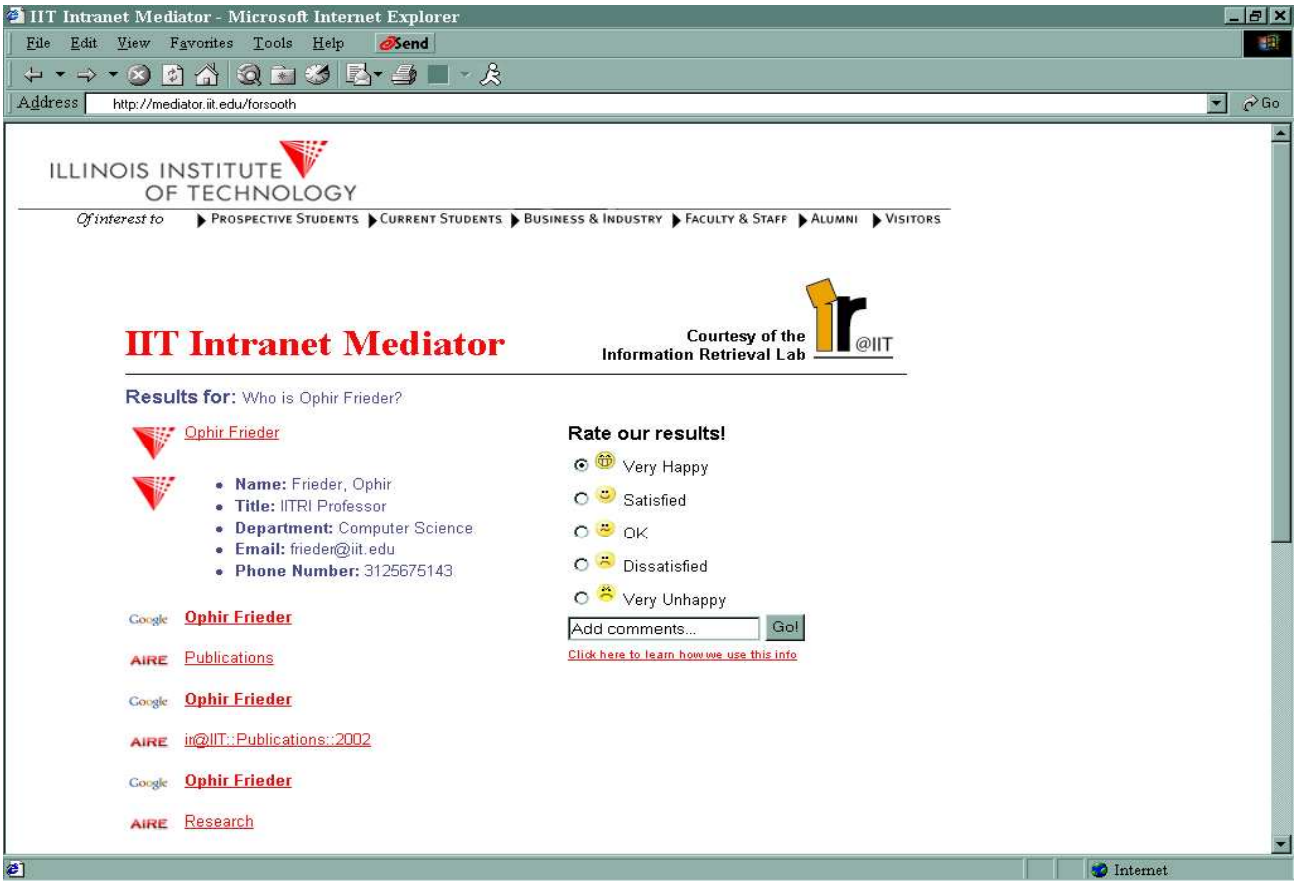


**Figure 5: Sample Output Screen**

Using a relational approach to information retrieval has multiple advantages since relational database systems provide enhanced functionality such as security, concurrency control, recovery, and parallel scalability without requiring the re-development of such features for the information retrieval domain. In fact, in terms of scalability, major relational vendors are under tremendous pressure to ensure that their implementation is scalable, hence supports a parallel implementation. The TPC-C and TPC-D benchmarks increase this pressure as system performance is measured in a publicly available forum.

Granted these benchmarks are short banking transactions rather than information retrieval queries, but the point is that significant engineering work is done in the relational arena to provide good scalability. Our prior experiments with this approach have shown a speedup of 22-fold on an NCR machine configured with 24 processors [7]. Those efforts were in joint collaboration with NCR. Additionally, efforts at the University of Tokyo [8] demonstrated further scalability using the SIRE approach on a 100+ node PC cluster, and work at ETH-Zurich has shown that the SIRE approach can be used to improve throughput for document insertion and update as well as simple retrieval [9]. NCR currently relies on an approach based on SIRE for some of their commercial text processing solutions.

## 3.2. Intranet Mediator

A mediator, an integration fabric for multi-source, multi-format data, sits between users and data. In response to a natural language question, the mediator identifies appropriate sources to respond to the question, accumulates the results, and ultimately ranks and transmits them to the user. A search engine resembles a mediator, but a search engine strictly focuses only on unstructured data, typically text. In reality, however, there are three broad types of data: structured (e.g.; data stored in conventional relational database systems), unstructured (text, video, audio), and semi-structured (e.g.; XML). The IIT mediator is designed to interact with all three high-level data-types – not simply unstructured data. The high-level mediator architecture is given in Figure 4.

The mediator consists of the following components:

***Query Processor:*** The query processor takes a natural language query and parses it into key grammatical constructs such as *subject*, *verb*, and *objects*.

Additionally, the query processor performs a part-of-speech tagging operation on the query to identify the most likely part of speech for each term in the query. Finally, an entity tagger is used to identify top-level semantic concepts, such as *location, person, place, organization, etc.* in the query.

***Level 0 Rules:*** These rules take the syntactic elements in the query and existing metadata lists and identify higher-level semantic concepts in the query. Consider a course number like *"CS 522"*. This might be recognized by the two character prefix *"CS"*, and then the three digit sequence. A level 0 rule might be of the form: If *subject or object = [list of course prefixes]* [3 digits] then *subject or object = [COURSE_NUMBER]*.

***Level 1 Rules:*** 'Level 1' rules take output from the query processor and semantic concepts identified by level 0 rules and map to one or more *retrieval functions*, which are used to obtain the actual data that comprise the answer to the query.

***Retrieval Functions:*** These small functions contain the code needed to actually retrieve data from a source. These speak to the source in the language of the source – a relational source will have SQL, and an XML source might be sent XML-QL or some other XML query language. One might note that the whole game of retrieval from multiple heterogeneous sources is simply one of taking the English query and, from it, choosing the right retrieval functions. The idea is that the combined efforts of the rules framework (both level 0 and level 1 rules) will enable the selection of the correct retrieval functions.

***Dispatchers:*** The various source-type dispatchers handle the task of asynchronously invoking the appropriate retrieval functions for the sources that have been deemed appropriate to the query.

***Results Manager:*** Once the set of matching retrieval functions execute, their results are collected and submitted to the results manager component for unification, ranking, and duplicate removal. A sample output screen is given in Figure 5.

***Metadata Analyzers:*** The mediator also contains analyzers that examine new input sources and identify key aspects of the source. Currently our analyzers anticipate that a source is the actual data. This will likely be enhanced in the near future to accept a service that allows access to data rather than the data itself.

## 4. Summary

Scalable search systems require the integration of multi-type, geographically dispersed, highly-voluminous data repositories into a single unified information portal. We described several research efforts from the IIT Information Retrieval Laboratory that address some of the key issues associated with the design of scalable search systems. Some of those efforts described are now in daily use in several major commercial and governmental information processing organizations.

## 5. References


[1] R. Shuey, D. Spooner, and O. Frieder, The Architecture of Distributed Computer Systems: A Data Engineering Perspective, Addison Wesley, ISBN 0-201-55332-5, 1997.

[2] D. Grossman and O. Frieder, Information Retrieval: Algorithms and Heuristics, Kluwer Academic Publishers, ISBN 0-7923-8271-4, 1998.

[3] A. Chowdhury, O. Frieder, D. Grossman, and M. McCabe, "Collection Statistics for Fast Duplicate Document Detection," ACM Transactions on Information Systems (TOIS), 20(2), April 2002.

[4] D. Grossman, S. Beitzel, E. Jensen, and O. Frieder, "IIT Intranet Mediator: Bringing Data Together on a Corporate Intranet," *IEEE IT PRO*, January/February 2002.

[5] D. A. Grossman, O. Frieder, D. O. Holmes, and D. C. Roberts, "Integrating Structured Data and Text: A Relational Approach," *Journal of the American Society of Information Science*, 48(2), February 1997.

[6] O. Frieder, A. Chowdhury, D. Grossman, M. C. McCabe, "On the Integration of Structured Data and Text: *A Review of the SIRE Architecture," DELOS Workshop on Information Seeking, Searching, and Querying in Digital Libraries*, Zurich, Switzerland, December 2000.

[7] C. Lundquist, O. Frieder, D. Holmes, and D. Grossman, A Parallel Relational Database Management System Approach to Relevance Feedback in Information Retrieval. Journal of the American Society of Information Science, January 1999.

[8] K. Goda, T. Tamura, M. Kitsuregawa, A. Chowdhury, and O. Frieder, "Query Optimization for Vector Space Problems," *ACM Twenty-Fourth SIGIR*, New Orleans, Louisiana, September 2001.

[9] T. Grabs, K. Böhm, H.-J. Schek, "High-level Parallelisation in a Database Cluster: a Feasibility Study Using Document Services," *IEEE 17th International Conference on Data Engineering (ICDE2001), Heidelberg, Germany, April 2001.*