Retrieval Activities in a Database Consisting of Heterogeneous Collections of Structured Text

Forbes J. Burkowski

Department of Computer Science
University of Waterloo
Waterloo, Canada

Abstract

The first part of this paper briefly describes a mathematical framework (called the containment model) that provides the operations and data structures for a text dominated database with a hierarchical structure. The database is considered to be a hierarchical collection of contiguous extents each extent being a word, word phrase, text element or non-text element. The filter operations making up a search command are expressed in terms of containment criteria that specify whether a contiguous extent will be selected or rejected during a search. This formalism, comprised of the mathematical framework and its associated language, defines a conceptual layer upon which we can construct a well-defined higher level layer, specifically the user interface that serves to provide a level of functionality that is closer to the needs of the user and the application domain.

With the conceptual layer established, we go on to describe the design and implementation of a versatile interface which handles queries that search and navigate a heterogeneous collection of structured documents. Interface functionality is provided by a set of "worker" modules supported by an "environment" that is the same for all interfaces. The interface environment allows a worker to communicate with the underlying text retrieval engine using a well-defined command protocol that is based on a small set of filter operators. The overall design emphasizes: a) interface flexibility for a variety of search and browsing capabilities, b) the modular independence of the interface with respect to its underlying retrieval engine, and c) the advantages to be accrued by defining retrieval commands using operators that are part of a text algebra that provides a sound theoretical foundation for the database.

1. Introduction

It can be argued that the main strength of a relational database is the *relational model* at its foundation. As Date (1990, pg. 12) observes:

"Thanks to this solid foundation, relational systems behave in well-defined ways; and (possibly without consciously realizing the fact) users have a simple model of that behaviour in their mind, one that enables them to predict with confidence what the system will do in any given situation. There are (or should be) no surprises. This predictability means that user interfaces are easy to understand, document, teach, learn, use, and remember."

Recently a number of researchers have developed models for text databases. Research in this area includes Gonnet and Tompa (1987), Gyssens, Paredaens, and Van Gucht (1989), and Tague, Salminen, and McClellen (1991). More recently, Burkowski (1992) describes an algebra for hierarchical text dominated databases. In this model the database is considered to be a hierarchical collection of contiguous extents each extent being a word, word phrase, text element (eg. chapter, section, etc.) or non-text element (eg. bitmap) and the filter operations making up a search command are expressed in terms of containment criteria that specify whether a contiguous extent will be selected or rejected during a search. This *containment model* is meant to provide a foundation for structured text databases that is as "solid" as the relational model foundation provided for relational databases.

Text databases can utilize a variety of retrieval techniques. As described by van Rijsbergen (1979, pg. 2), systems characterized as providing data retrieval typically use Boolean queries to define exact match criteria while information retrieval systems employ probabilistic techniques that foster relevance feedback strategies. An interesting paper by Cooper (1983) distinguishes the use of faceted searching and coordination-level matching in query formulations. Some system designers utilize Boolean queries for databases in which probabilistic techniques are deemed to be inappropriate. For example, the Oxford English Dictionary project (Gonnet (1987)) employs Boolean queries for dictionary searches while Hoppe, et al. (1990) rely on Boolean oriented retrieval tactics and strategies for "factual information retrieval". McAlpine and Ingwersen (1989) use exact match Boolean logic in the EUROMATH project. Another interesting tactic is to use complex Boolean retrieval "behind" an interface that presents a more effective sophisticated yet friendly interaction with the user. Especially noteworthy is the work done by Anick, Flynn and Hansen (1991) who use "Query Reformulation Workspaces" to help overcome the traditional shortcomings of the Boolean query.

While many systems tend to be at one extreme (Data Retrieval) or the other (Information Retrieval) there is a functional middle ground that can benefit from the marriage of both types of query facilities. Sacks-Davis, Wallis and Wilkinson (1990) use an initial Boolean query to extract a subset of documents that are then ranked.

1.1 Objectives

This paper will describe the containment model and the mathematical framework (text hierarchy) in which it works. We then go on to demonstrate that typical Boolean queries are really *containment specifications.* This is done by presenting examples that illustrate the syntax of the Retrieval Command String (RCS) protocol between a user interface and its underlying (usually server resident) retrieval engine. This RCS facility is then adapted to provide statistical ranking of the search results thus extending the model into the IR application domain.

The primary purpose of this paper is to present a unifying model that can promote the synergistic cooperation of both Boolean and statistical ranking methodologies. We then show how the model can be utilized by describing a modular, flexible and expandable user interface that interacts with the underlying retrieval engine by using the Retrieval Command Strings defined earlier. Readers may wish to contrast this approach with other client/server protocols such as SFQL specified by the ATA/AIA Subcommittee (1990) and Z39.50 prepared by NISO (1989).

Since the containment model assumes the existence of a structured text hierarchy we start by defining the nature of such a database.

1.2 Structured Text

Recent trends have demonstrated a rapid increase in the generation of text collections comprised of structured documents that incorporate descriptive markup to define both the logical model and the layout model of its constituents. While many text collections use customized or in-house markup, there has been a definite shift toward the use of markup facilities provided by popular standards such as SGML and ODA (described in reports from the International Standards Organization, (1986) and (1988) respectively). As the acceptance of standardized markup spreads we can expect even greater quantities of structured on-line text with a concomitant demand to have retrieval operations utilize and react to the logical tags and layout tags embedded in the text.

Typical applications include storage and retrieval of reference material, encyclopedias, legal precedents, educational presentations, books, and journal publications. Raymond (1991) describes the use of text databases for retention and retrieval of source code.

1.3 Characteristics of the Structured Text Database

The following points will describe those features of a structured text database that are germane to the scope of this paper:

1) Heterogeneity

   The tagging scheme may differ across the various constituents of the database. For example, if the database includes the complete works of Shakespeare we would expect the collection of plays to have a structure that is different from that of the collection of sonnets.

2) Multi-media

   Even though retrieval activities typically rely on the database being text dominated, presentation of the data may include images and sound.

3) Hierarchical Layers

   The tags embedded in the text can have more than one purpose. They may determine the logical structure of the text, for example, delimiting the nested extents that represent chapters, sections, subsections, etc. or they may specify the layout (presentation formats) of the text when it is displayed. Although both sets of tags coexist in the same text they each define a hierarchy of nested extents in which there is no partial overlap. For example, two extents of the logical structure will be either disjoint or one will be nested in the other. However, there may be overlap of extents across the two hierarchies, for example, a page from the layout hierarchy may partially overlap two consecutive chapters of the logical hierarchy.

As noted by Meghini, Rabitti and Thanos (1991), it is highly advantageous to have another set of tags that define a *semantic* layer for the text. This layer (which may involve multiple levels with a hierarchical organization) serves to give a text extent some particular meaning that goes beyond its word content. For example, the tags in

<city>Washington</city>

clearly identify this single word text extent as the name of a city. Another approach has been promoted by Rau and Jacobs (1991) using a method called segmented databases. The keywords are divided into segments that create *conceptual categories* of keywords (for example, a 'company_name' segment).

1.4 Design Goals of the Text Retrieval System

The following list defines some of the more important capabilities that should be expected in a database that contains heterogeneous text collections:

1) Data Independence

   The user interface should not be required to know how the text collection and its index facilities are managed. Ideally, it will accomplish retrieval activities by dealing with a conceptual model of the database expressing search operands using symbols that are mapped to their physical associations through the use of tables and indexes that are retained by an underlying retrieval engine. A major goal is to isolate the complexity of the retrieval engine by encapsulating it in a module that will respond to simple command strings.

2) Utilization of the Various Layers of the Hierarchy

   Searching and browsing of the database can be considerably enhanced when the logical, semantic and layout hierarchies are properly utilized. Browsing of the text hierarchy can be accomplished using a "table of contents" metaphor if the logical hierarchy identifies the titles of nested text elements such as parts, chapters, sections, etc. Queries involving a search for specific terms will have more precision if the semantic hierarchy is used to provide meaningful containment constraints. Thus, if the <city> </city> tags are used in the text, a search for "Washington" the city will not fetch "Washington" the statesman.

3) Extensible Interface Functionality

   The user interface for a text retrieval system should be easily adapted (by the system designer) to cover a variety of retrieval activities with the possibility of rapidly customizing particular navigation and search techniques that are appropriate to a specific data collection or user group.

The attainment of these objectives involves the proper management of complex issues. The heterogeneity of the data collections will complicate the user interface since search and navigation techniques can vary from one data collection to the next. There are two reasons for this:

a) the tags will differ in different data collections,

b) the "fanout" of the hierarchy can vary from one data collection to the next. For example, a "table of contents" type of browse activity may easily descend the hierarchy if the data collection deals with books. A chapter will contain, at most, a few dozen sections that can be listed by title in the user interface. The situation is very different if the local hierarchy of the database leads to a newspaper collection. It is not reasonable to start listing titles of thousands of newspaper articles and so it would be advisable at this point to continue retrieval activities by initiating some other strategy such as a full-text search.

2. An Algebra for Contiguous Text Extents

2.1 Motivation

One of the principle assertions in this paper is that data independence, system modularity and integrity of the database design will be greatly enhanced if the user interface communicates with the retrieval engine by using a small set of commands based on the rigorously defined text algebra of the containment model. Such an algebra, described by the author (1992) will be summarized in section 2.2.

Before describing the algebra we define the organization of the database. It will have a hierarchical structure that organizes the text into various groupings that we will call static contiguous extents. A **contiguous extent** will be a single word, a phrase comprised of consecutive words, or some longer sequence of consecutive words referred to as a text element. A **text element** is a sequence of contiguous words that has a particular significance in the database. For example, if the database is comprised of newspaper articles, the text elements might include: headline, dateline, textbody, date, or author. Contiguous extents may be static (defined during database creation) or dynamic (defined during retrieval operations). Text elements may be defined either implicitly (and subsequently located during the database load using a parser) or explicitly using markup with tags.

We assume the database is created by a LOAD activity that defines the initial text hierarchy by generating a set of **concordance lists** that keep track of the position and nesting properties of the various static contiguous extents such as words and text elements. After the load activity is completed each concordance list will identify all instances of a particular type of contiguous extent. For example, one list could specify all instances of the 'chapter' text element while another list might specify all instances of the word "thunder". The concordance lists will completely define the hierarchical structure and data content of the database. The text algebra provides filter operators that can be used to manipulate these lists in response to the needs of a query. A parser is used to transform a user query into a sequence of filter operations that are submitted to the retrieval engine. In the retrieval engine various concordance lists are used as operands of the operators that on evaluation produce new concordance lists. The final concordance list that is computed will designate the contiguous extents that satisfy the requirements of the query. The operators and concordance lists essentially define an abstraction that we can regard as a **conceptual level** of the database architecture. The conceptual level captures the main essentials of the text hierarchy defining both the containment properties of its contiguous extents and the order of these extents within a larger extent. The user view of the database may conform to this conceptual view of the database or it may be quite different depending on the facilities provided by the user interface.

In this fashion the conceptual level provides a measure of **data independence**. In practice we have found that changing the internal functionality of the retrieval engine will not lead to modifications in the user interface as long as the new engine supports the same search operators. Conversely, a change in the interface will not require a new version of the engine; a new interface issues a different sequence of retrieval operations to the same retrieval engine.

We will define filter operations based on the containment model. They will operate on the concordance lists defined during the database load. We contend that there are advantages in adopting the mind set that the full concordance *is* "the database" (it is, of course, the inverted form of the database). The text collection itself will be accessed only when a "fetch" operator is initiated. These advantages arise from the ability to precisely formulate a query specification in terms of the fundamental retrieval operations employed by the database server.

In summary, utilization of the containment model has a two fold advantage:

a) it provides a rigorous foundation for the specification of retrieval activities, and

b) filter operators defined on the model form the basis for a relatively simple but powerful protocol between the user interface and the retrieval engine.

2.2 A Formal Definition

A **text collection** is a finite sequence of words w_0, w_1, w_2, ... w_{n-1} where each $w_i \in V$ a set called the **vocabulary** of the text collection.

A **contiguous extent** e is specified by two integers $\alpha(e)$ and $\omega(e)$ such that $0 \le \alpha(e) < \omega(e) \le n$. We define e to be the text collection subsequence that includes all words w_i such that $\alpha(e) \le i < \omega(e)$, that is, $i \in [\alpha(e), \omega(e))$. We will refer to the pair of integers $[\alpha(e), \omega(e)]$ as the **bounds** that specify the contiguous extent e. Note that the simplest contiguous extent would be a single word w_i specified by the bounds [i, i+1].

For two arbitrary extents x and y, we say that x is **nested** in y denoted by $x \subset y$ if x specifies an extent that is totally included within the extent represented by y. That is:

$$x \subset y \iff \alpha(y) \le \alpha(x) < \omega(x) \le \omega(y).$$

Similarly,

$$x \supset y \iff \alpha(x) \le \alpha(y) < \omega(y) \le \omega(x).$$

If two contiguous extents x and y are such that either $\alpha(x) \ge \omega(y)$ or $\alpha(y) \ge \omega(x)$ then x and y are said to be **disjoint**. Depending on context, the terms nested and disjoint will also be used in relation to the bounds of contiguous extents.

A **concordance list** $G = \{[\alpha(e_i), \omega(e_i)]\}^m_{i=1}$ is defined to be a named list of bounds specifying disjoint contiguous extents. G is the **name** of the concordance list and m is its **cardinality**. In an application, the contiguous extents delimited by these bounds will typically share some similar property. For example, each contiguous extent might be designated as a "chapter" in the text collection.

Database retrieval will be done using one or more **Retrieval Command Strings**. Each RCS will be expressed as a sequence of **filter** operations. There are two types of filter operations: **select** and **reject**. Each of these types can be viewed as **narrow** or **wide** depending on the containment criteria specified in the definition of the filter operations:

Select Narrow:

$$A \ \mathbf{SN} \ \{B_1, B_2, B_3, \ldots B_q\}$$

The result C of this operation will be a subset of the list A. For $a_i \in A$, the entry a_i is selected, that is $a_i \in C$

if and only if $a_i \subset b_{mn} \in B_m$ for some $1 \leq n \leq |B_m|$ and for some $1 \leq m \leq q$.

Select Wide:

$$A \ \mathbf{SW} \ \{B_1, B_2, B_3, \ldots B_q\}$$

The definition for **SW** is the same as that for **SN** except that $\subset$ is replaced by $\supset$.

Reject Narrow:

$$A \ \mathbf{RN} \ \{B_1, B_2, B_3, \ldots B_q\}$$

The result C of this operation will be all members of A except those that are rejected. For $a_i \in A$, a_i is rejected that is $a_i \notin C$ if and only if there are q integers k_{im} $m = 1,2, \ldots ,q$ such that:

$$1 \leq k_{im} \leq |B_m| \quad m = 1,2, \ldots ,q$$

and $a_i \subset b_{mk_{im}} \in B_m$ for all $m = 1,2, \ldots ,q$.

Reject Wide:

$$A \ \mathbf{RW} \ \{B_1, B_2, B_3, \ldots B_q\}$$

The definition for **RW** is the same as that for **RN** except that $\subset$ is replaced by $\supset$.

2.3 Some Examples

In the examples to be presented, the list identifiers will be represented symbolically using either a single word, phrase structure, tag or list name (identifying the result list of a previous search).

Examples of these would be:

a) "proton"

This is a symbolic representation of the concordance list that specifies the word positions of each appearance of the word "proton" in the database.

b) "tensor calculus"

This is the representation of a concordance list of bounds of contiguous extents each one being the two word phrase "tensor" immediately followed by "calculus",

c) <chapter>

This syntax would represent the concordance list that retains the bounds of all contiguous extents that have been tagged as a 'chapter'.

d) result_list

We use a single variable name to designate the name of a concordance list that has been dynamically generated during a retrieval activity. These lists can be used in a future operation, for example, in the refining of a search.

By specifying a search activity the user essentially tells the system *what* a fetched text element (such as a document) should contain and *where* it should contain it. We will now illustrate this use of the filter operators by reviewing some examples similar to those presented in Burkowski (1992).

A search for chapters that contains "Feynman" OR "virtual particle" would be:

<chapter> **SW** {"Feynman", "virtual particle"}

while a search for "Feynman" AND "virtual particle" would be:

<chapter> **SW** {"Feynman"} **SW** {"virtual particle"}.

Searching for chapters with titles that contain "weak photons" OR "Higgs vacuum" would be:

<chapter> **SW** {<chapter_title> **SW**
{"weak photons", "Higgs vacuum"}}.

The filter operator **SW** is called a **Select Wide** since it selects elements that contain other specified contiguous extents.

The RCS (Retrieval Command String) can also permit a type of search that generates the bounds of contained extents rather than bounds of the elements that contain them. Thus,

<section_title> **SN** {chapter_list}

will develop a concordance list specifying text extents each being an element that is tagged as a title of a section and each contained in a text element with bounds in the chapter_list concordance list presumably defined by an earlier query. This filter operation is called **Select Narrow** since it selects contiguous extents that are contained within other specified extents.

We may use a type of **disqualification** filter that serves to reject text elements, for example:

<chapter> **SW** {"hadron"} **RW** {"electron", "neutrino"}

We assume execution of the operations in left to right order since there are no parentheses to alter this order. This RCS will select all chapter text elements containing "hadron" but will subsequently reject (**Reject Wide**) any text element that contains *both* "electron" AND "neutrino".

We see that the filter operations making up a search activity all deal with *containment rules* that define whether some contiguous extent will be selected or rejected during a search.

The power and flexibility of a RCS originates from a design philosophy that does not attempt to distinguish the *types* of contiguous extents involved in a search. These contiguous extents will be treated differently *after* a search, when and if they are fetched and displayed or printed.

2.4 More RCS Syntax

We now present more RCS syntax that supplements the filter operations just described.

Specifying results of a previous search:

When a RCS is executed the retrieval engine does the indicated list manipulations to produce a final result that may be named for future reference. Thus, in the example above, the variable name 'result_list' can be used later to reference the final filtered list. This list stays with the retrieval engine. It is not returned to the interface since the interface would not have the capability of handling it (recall the data independence goal).

Fetching contiguous extents:

Fetching a text extent is done using the [.] operator. Thus,

result_list[3]

will return the document that is the text element specified by the result_list entry with index 3.

Cardinality of a search result:

The return value for an RCS will be its cardinality, a value representing the number of entries in the final list. We will refer to this return value using vertical lines, for example, |result_list|.

Sublists of concordance lists:

At various times it will be necessary to extract list entries from a previously generated list in order to form a sublist. Thus, result_list(5) represents a new list with a single entry, namely the entry with index 5 in result_list. A range of entries would be extracted using two values in the brackets, that is, (m:n).

Word length of a contiguous extent:

When the length of a text extent is required, we use the **LENGTH** function. For example:

LENGTH{result_list(2)}

will return the word length of the text extent that would be fetched using the operation result_list[2]. Considering the α, ω notation defined earlier, this length will be ω(result_list(2)) - α(result_list(2)).

2.5 Statistical Ranking

The containment model adopts statistical ranking of documents (more generally, text extents) as an activity that follows an initial extraction of these text extents using a "Boolean-like subquery". This is similar to the previously mentioned strategy used by Sacks-Davis et al. (1990). It is also analogous to the approach used in relational databases. A retrieval language such as SQL will extract a relation (this extraction being formally defined via the relational algebra) and after this extraction, sorting of the data can be undertaken if required (note that, sorting of retrieved results is *not* defined by the primitives of the relational algebra).

Since the database is a heterogeneous collection of various document types, an initial subquery extraction of text extents is both natural and necessary. Typically, the initial extraction will pull out text elements with similar substructure or (perhaps more appropriately) with a similar semantic context.

We use a statistical ranking strategy similar to that described by Harman (1990). Ranking is done with an Inverse Document Frequency (IDF) weighting measure. The weight of an extracted document with index j will be:

$$\sum_{k=1}^{Q} \frac{(\log_2 Freq_{jk} \times IDF_k)}{\log_2 M_j}$$

where:

Q	=	the number of terms in the query,
$Freq_{jk}$	=	the frequency of query term term_k in document j
M_j	=	the total number of words in the document j (the document length)
IDF_k	=	the inverse document frequency weight of term_k
	=	$\log_2\left(\frac{N}{NumD_k}\right) + 1$

where:

N	=	the number of documents of this type in the database
$NumD_k$	=	the number of documents in the collection that contain one or more instances of term_k

In the following discussion let us assume:

a) the query terms are designated as term_1, term_2, ..., term_k, ...,

b) the entire database is delimited with the tags <db> and </db>,

c) the initial subquery has filtered all text extents tagged with <doc> and </doc> to produce a result list named 'extracted_docs'.

Under these assumptions we can use the RCS syntax to restate the definitions introduced earlier:

Q	=	the number of terms in the query,
$Freq_{jk}$	=	|term_k **SN** {extracted_docs(j)}|
M_j	=	**LENGTH**{extracted_docs(j)}
N	=	|<doc> **SN** {<db>}|
$NumD_k$	=	|<doc> **SW** {term_k}|

Note that the containment model utilizes a dynamic evaluation of both $Freq_{jk}$ and IDF_k. There are some advantages to this approach:

1) The system does not maintain frequency counts for individual words of the database vocabulary. These are evaluated dynamically. Since the evaluations of $Freq_{jk}$ and IDF_k are expressed in terms of text extents, the system can deal with single words or phrases in the query. Phrases involve a few more disk accesses, but once the phrase text extents are known the calculations proceed just as for single words. This simple approach considers each text extent in a query to be a "semantic unit" and so avoids the need to estimate the weight of a phrase using the weights of its constituent words as done in Fagan (1987). It also avoids phrase related correction factors derived from a dependency assumption for phrase constituents as discussed by Croft (1990).

2) Since a heterogeneous database will contain sub-hierarchies with a wide variation in the logical structure, it is difficult at system design time to predict those types of text extents that should be involved in the ranking process. The design philosophy presented here is that the retrieval engine should provide the necessary environment so that any future interface can specify, during retrieval, a ranking procedure that utilizes the appropriate text structures.

Although the containment model provides great flexibility in the choice of text extent types that are to be involved in ranking, it does not give any specific indication as to which type of text extent is best utilized. These are issues for the system designer and may well depend on both the nature of the application and the logical structure of the local hierarchy. The reader may consult Wendlandt and Driscoll (1991) who have compared ranking strategies related to paragraphs and sentences. Salton and Buckley (1990 and 1991) have also experimented with text matching systems that operated at various levels of text granularity.

Ranking evaluations are best done within the retrieval engine to avoid the transmission of long vectors between the workstation interface and the server. Consequently, the interface requests a ranking evaluation by issuing a function call to the retrieval engine, for example:

ranking_vector = **RANK**(extracted_docs, <doc>,
term_1, term_2, ... ,term_k, ...)

where the arguments have the same significance described earlier in this section.

The retrieval engine does the weight evaluation, sorts the weights and sets up an index permutation that is given the name on the left hand side of the assignment. This name can be used in conjunction with the result_list to pull out ranked documents. For example,

```
        extracted_docs[ranking_vector(0)]
```

returns the highest ranked extracted document while

```
best_docs = extracted_docs(ranking_vector(0:4))
```

generates a list (named 'best_docs') containing the bounds from the extracted_docs concordance list specifying the five extracted documents of highest rank.

3. Database Structure and Interface Organization

3.1 Global Structure of the Database

We now describe a system that meets the objectives described earlier. This database design has been developed by the author and is representative of the type of system that could be supported by the previously described containment model and filter operators. Undoubtedly, there are other designs that would also be effective while still conforming to the basic hierarchical structure of the database as required by the model.

Figure 1 presents the macroscopic view of the database. The entire database is delimited by the <db> </db> tags. Immediately descendent from this topmost node of the hierarchy is a set of nodes each defined to be a **data collection**. Each data collection will typically have a different structure. This will accommodate the required heterogeneity of the database. Subsidiary to each data collection node is a set of text extents:

a) the pathname of the Tag and Hierarchy Specification (THS) file,

b) the title of the data collection, and

c) various local hierarchies.

Each local hierarchy has a similar structure that is specified by the contents of the sibling THS file.

3.2 THS (Tag and Hierarchy Specification) File

The THS file specifies the various properties associated with a sibling local hierarchy. These include the strings to be used as interface menu items that refer to a text element, the formats of tags appearing in a RCS, and a specification describing the structure of the local hierarchy. The structure is described by specifying two sets of text element names:

1) The names of text elements in the "spinal sequence" of the hierarchy.

   The spinal sequence is the main sequence of nested text elements for this local hierarchy. For example, if the local hierarchy contains a collection of books categorized under various subject headings, we could have the following spinal sequence: subject, book, chapter, section, subsection.

2) The names of secondary extents.

   Secondary extents are tagged extents that may be used in a full - text search or cross reference activity. They are not part of the spinal sequence. An example might be a text extent that is semantically tagged to indicate that it is the name of a city.

3.3 Interface Organization

The interface consists of an "environment" and a collection of "Workers" each one associated with a window in the display. The environment consists of all the modules and facilities that are common to every interface, for example:

a) modules that accomplish system initialization, and

b) the communication facility that allows a worker to issue commands to the retrieval engine and then subsequently receive responses from the engine.

The workers include a collection of **prototypical workers** that may be modified or used "as-is" in the required interface (this includes a specialized Browser called the Data Collection window to be described later). There is also a collection of window items (edit boxes, list boxes, etc.) that can be used to develop other workers.

This organization of the interface promotes the flexibility and extensibility described earlier. Typically, a new worker can be added by modifying a copy of one of the existing workers. A precise set of procedures guide the system designer in this effort.

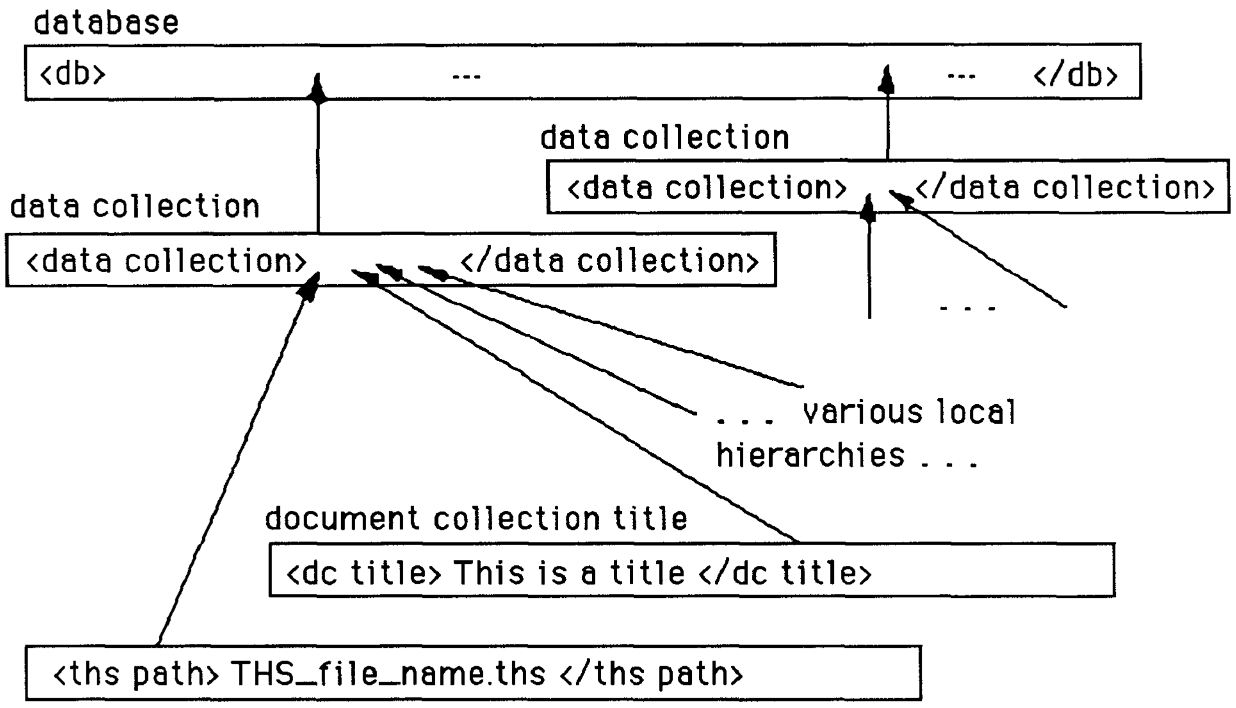
database
<db> --- --- </db>
data collection
<data collection> </data collection>
data collection
<data collection> </data collection>
. . .
. . . various local hierarchies . . .
document collection title
<dc title> This is a title </dc title>
<ths path> THS_file_name.ths </ths path>

Fig. 1: Global Structure of the Database

3.4 Interface Functionality

During execution the first worker launched by the interface is the Data Collection window that provides the starting point for all subsequent retrieval activities. This window will display the titles of the various data collections. Titles appear in a list box and selection of a title will allow a hierarchical descent into that data collection through the use of a Browser window. Further descent may or may not be permitted depending on the "fanout" structure of the local hierarchy. Menu items in the Browser can be used to initiate a full-text search constrained to a range that has been selected in the Browser list box. Since any worker will have menu items that cause the launch of other workers a chain of workers can be developed to carry out retrieval operations involving further navigation and searches of the database.

Workers other than the Data Collection presentation may vary in functionality depending on the needs of the local hierarchy. The prototypical workers can be segregated into some basic categories that include the following:

Full-Text Search workers
Browsers
Cross Reference workers
Display windows

4. The Worker Paradigm

As illustrated in Figure 2 the typical worker communicates with a variety of other components in the system. These will now be described.

4.1 Input and Output Facilities for a Worker

Each worker has the following input requirements:

a) A work order issued as a work request by another worker

b) Input from the user
   - text in edit boxes
   - list box selections
   - menu selections
   - button activations

c) Responses from the retrieval engine
   - hit count for the last search command (cardinality of a result list)
   - text extents extracted from the database

Each worker has the following output capabilities:

a) A work request issued to another worker

b) Output to the user
   - text in various output areas
   - text in list boxes
   - menu and button enabling and disabling

c) Commands to the search engine
   - Retrieval Command Strings (search activity specifications, requests for text extents, ranking requests).

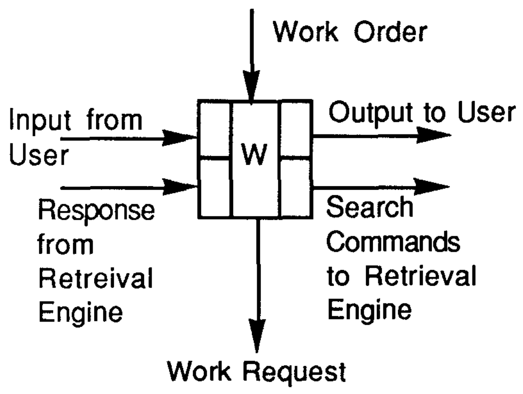
Work Order
Input from User
W
Output to User
Response from Retreival Engine
Search Commands to Retrieval Engine
Work Request

Fig. 2: Input and Output Facilities for a Worker

4.2 Work Orders / Requests

A work order / request consists of one or more of the following:

a)	Search constraints	- for full - text searches and cross references
b)	WITHIN constraints	- for a browse activity
c)	Schema definitions	
d)	Miscellaneous text	- title of a text element, captions, etc.

5. Worker Examples

As indicated previously, the Retrieval Command Strings are a well defined protocol for communication between a worker and the retrieval engine. Typically, the interface worker will screen the user from the complexity of RCS generation offering him or her a friendly easy-to-understand query generation facility. While many of the current commercial user interfaces provide "standard" Boolean queries, it should be stressed that a wide variety of creative and novel possibilities can be implemented. A good example of such an interface is in the work of Thompson and Croft (1989). It employs browse facilities and full-text searches with both Boolean query and probabilistic models.

To keep the examples simple (due to space limitations) we will describe a fairly "generic" text retrieval interface discussing both its functionality and the RCS sequences that it generates. This will illustrate the flexibility and power of the Retrieval Command Strings.

The interface relies heavily on the hierarchical structure of the text in the database. We will describe two of the worker prototypes:

a) The **Browser** is used for navigation "up and down" a local hierarchy. The main component of the Browser is its listbox which is used to display the titles of text elements that are subsidiary to some previously selected text element. This interface functionality can be given an intuitive feel by employing a "Table of Contents" type of metaphor. Choices of text elements are provided by means of miniatures (usually titles) that are fetched and placed into a list box. Individual items are subsequently selected by using a mouse operation. If a title is selected (single click) the parent text element may be involved in either one of two subsequent actions initiated via the menu: a full-text search that is limited to the selected text element or a display of the text element in a text window. Double-clicking a title will cause a "local expansion" beneath that title and the titles of subsidiary text elements will be displayed. The double-clicking functionality is actually a toggle type of functionality. If a title has already been expanded, a further double-clicking action will eliminate the titles produced by the earlier local expansion.

b) The **Full-Text Search** worker is used to develop lists of text elements each satisfying a Boolean constraint. These text elements are typically spread across a very wide horizontal range of the hierarchy. Full-text searching can be done any time, for example, when a Browser descent in the hierarchy is essentially impossible due to a large "fan out" of subsidiary text elements.

5.1 RCS Commands Issued by the Browser

Browsing activity is typically initiated at the beginning of a user session when a data collection is selected from the main application window. After a selection is made a further Browser descent may or may not be effective depending on the nature of the data collection (specifically, the "width" of the hierarchy below the current text element). If a descent is possible, a Browser window is created and choices of elements at the upper level of the local hierarchy are made available in a list box that is part of the Browser window.

As an example, consider the command:

```
listbox_items  = <chapter_title>SN
                        {<chapter>SN{<manual>}}
```

that generates a list of the bounds of required chapter titles that will provide the initial contents of the list box. The right-most SN is evaluated to derive a list of all chapter elements in a "Manual" subhierarchy (delimited with tags <manual> and </manual>). The first **SN** is then evaluated to extract the titles of these chapters. The engine returns a hit count specifying the number of titles that have been found (for example, 8). These are subsequently fetched using the invocation:

```
        listbox_items[0:7]
```

and are inserted into the list box.

When a list box item is double-clicked, say number 5, (at index 4) the parent text element of the title is obtained using a **SW** and the subsidiary text elements are derived using a **SN**. The titles may then be extracted using a further SN. The compound search will have the form:

```
new_items = <section_title> SN
    {<section>SN{<chapter>SW{listbox_items(4)}}}.
```

When a list box item is selected (at, for example, index 3) the contents of the parent extent may be presented in a display window by the execution of a search wide to find the bounds of the text element followed by a [.] operation to retrieve the text itself:

```
        <section> SW {new_items(3)}[0].
```

5.2 RCS Commands Issued by the Full-Text Search Window

The Full-Text Search window may be launched when the database is initially chosen. If a narrower search is required it can be launched from a Browser that has been used to descend to some particular text element of the database. The subsequent full-text search would be constrained or "clipped" to this (typically very large) text element.

In our implementations the FTS window consists of three sections: an area that is used to formulate a Boolean constraint, a hit count report, and a list box (similar to that appearing in a Browser) that is used to report the results of a search. The user will "fill in the blanks" in order to specify a Boolean constraint for a search. This Boolean condition will be satisfied *within* a text element that is chosen by a selection from the Within menu.

As an example, the RCS sequence sent out by the FTS invocation might be:

```
within_extents = <p> SW {"ozone depletion",
        "chloroflourocarbons"}SW {"global warming"}
```

which derives a list of the bounds of text elements (paragraphs) where the Boolean condition is satisfied. In this case 'Paragraph' was selected from the Within menu. This command is somewhat modified in the case of a constrained search launched from a Browser. In this case the list of <p> extents (paragraphs) can be clipped to those that are subsidiary to a pre-specified extent by using a search narrow.

The next command:

```
listbox_items = <article_title> SN
                {<article> SW {within_extents}}
```

derives the list of bounds of titles that will be established in the list box. Note that the **SW** will admit only those articles that contain the paragraphs derived by the Boolean condition. Once a list box item is selected, the actual text can be displayed using RCS interactions that are similar to those described earlier for the Browser.

When a local expansion is done in the listbox (say at offset 6) the compound RCS will be similar to that seen above in the Browser listbox expansion, but the subsidiary extents must contain the within_extents previously derived:

```
new_items  =  <section_title>SN{<section>SN
                    {<article>SW{listbox_items(7)}}
                            SW{within_extents}}.
```

A very important design decision of this interface was to use the table of contents metaphor when presenting the user with a list of database hits in response to a full-text search. When a search is completed the user is given a list of titles and the opportunity to investigate substructure using the same interaction as that provided by the Browser. This time however, when a title appears in the list box, it is only because the parent text element contains a subextent that satisfies the query.

This is a very convenient and effective way to investigate the hits produced by the search. Many text retrieval systems will present the results of a search as a lengthy collection of lines each containing the phrase or word used in the query. Even for simple one word queries, a single text line containing the query word or phrase will typically not provide enough meaningful information to give the user a worthwhile indication about whether this hit is really of interest. If the query is more complicated than a simple search for one word or phrase, the presentation of a single line will not be large enough to indicate the success of the search. By allowing the user to descend the hierarchy using a table of contents descent that is constrained to the hits, the user is given the equivalent of a "path name" to a hit and so he or she knows how the information has been categorized in the database. There is a benefit to this strategy. During the descent the user can avoid the investigation of hits that are obviously in areas of the database that are of little or no interest. These can be avoided simply by knowing the titles associated with their higher level structure. The elimination of retrieval of unwanted documents will reduce the cost of a search for a user and will lower traffic on a network that is providing the service. While we have not yet done any experiments that would compare the effectiveness of this retrieval strategy with other methodologies, we have received favourable reports from users who appreciate the ability to rapidly home in on the required text extents.

5.3 Workers and Hierarchical Layers

As mentioned earlier in section 1.1, the text tags may define one, two or three superimposed hierarchies (logical, layout and semantic layers). It is interesting to note the relationship between worker functionality and these various hierarchies. In the system just described a worker will utilize one or more layers as follows:

1) The Browser relies on the logical hierarchy providing navigation by means of search activities that deal with text elements and their associated titles.

2) The Full-Text Search window may specify containment criteria that are dependent on logical structure, for example,

   <chapter> SW {"behemoth", "leviathan"}

   but, as noted earlier, containment criteria dealing with the semantic layer typically provide more precise search results.

3) Cross reference workers may deal with any one of the three layers. Thus, a cross reference may specify a title or in some cases it may specify a page in the layout hierarchy.

4) Display workers will make use of the logical layer to localize the text element that is to be displayed while the layout hierarchy may be used to specify the presentation format.

5.4 Workers and the THS File

Now that two examples of worker prototypes have been described, the role of the THS is more apparent. Information in the THS file can serve to specify the parameters needed to initialize a worker when it is launched by a menu selection made in an earlier window. The THS file will establish the main sequence of nested text elements and their associated tags so that the Browser can issue the proper Retrieval Command Strings during its navigation of the hierarchy. Similar information is used by a Full-Text Search Window, in particular, it must be informed of all tags used in the semantic layer.

5.5 Managers

Since the launching of a worker invariably requires a work order issued as a work request by an earlier worker, we can imagine the workers to be linked by communication lines forming an invocation tree. The invocation tree is actually a simplification of the information flow between workers. In practice we use a **manager** that acts as a relay, accepting work requests from workers and sending them off as work orders to newly created workers.

A manager is first created when a descent is made from the top of the hierarchy (the Data Collection Choices window). This initial descent will be into a selected local hierarchy that will have a particular THS associated with it. Thus, each manager can be given a specification of the particular local hierarchy that is its domain. It can then communicate this information to any worker that it creates by augmenting a work order with the schema for the local hierarchy.

Other duties of the manager:

1) The manager will keep track of all active workers in case they must be destroyed at a later date.

2) The manager will have all the necessary parameters required for the instantiation of a worker (size of window, contents of the menu bar, etc.). This avoids the need for each worker to know the instantiation parameters of all the other workers.

3) The manager can arrange the placement of worker windows on the screen for the best viewing.

6. Future Research

We intend to investigate the following topics:

a) Data Security

   It seems likely that the filter operators can be utilized to provide data protection, for example, not allowing a specified class of users access to sensitive or proprietary information.

b) Stored Queries and Views

The filter operators should allow the support of stored queries and views. This would be similar to the facilities described by Anick et al. (1991).

c) A Retrieval Language

Development of the RCS protocol began with search activity specifications that were later extended to more comprehensive commands for operations such as ranking of text elements. However, the sending of a single line RCS is really a protocol limitation. In the case of ranking it essentially requires that the interface initiate a server process that always implements the same statistical ranking formula. While the types of text extents involved in the calculation can be specified, the formula itself cannot be changed unless it is reprogrammed in the server software. This could be avoided if the retrieval engine was to accept from the interface a program segment that it would interpret whenever an operation such as ranking was to be done. We are working on this extension.

7. Conclusions

This paper has described a hierarchical text database, its user interface and the command protocol between this interface and the retrieval engine. These strategies represent a practical utilization of the text algebra defined by Burkowski (1992).

Database Structure

We started by describing a database system that supports retrieval operations for a heterogeneous collection of text with a hierarchical organization. The hierarchical structure is defined either implicitly using some type of parsing facility or explicitly using markup with tags. A load activity creates a full concordance for all static text extents of the database (the words and all higher level text elements).

Filter operations based on a containment model operate on the concordance lists defined in this load.

A Conceptual Layer

The containment model represents an elegant and utilitarian foundation for many current text retrieval models. Its algebraic operations give us the benefits of a rigorous foundation for the specification of retrieval activities, and operations within this mathematical framework form the basis for a relatively simple command protocol between the user interface and a retrieval engine that may be on the same machine or on a database server accessible over a LAN.

We have shown that this command protocol can encompass the standard Boolean operations encountered when doing a full-text search or a browser navigation. Furthermore, the model can be extended to accommodate statistical ranking of text elements thus providing a framework for conventional information retrieval.

The Interface Layer

Interface facilities are supplemented with a development toolkit that presents a modular environment based on a "worker" paradigm. System designers can use this flexible and easily adaptable environment to create new workers or to modify existing workers.

Workers rely on the underlying conceptual layer to gain access to an encapsulated retrieval engine that retains the database while isolating the interface from the internal complexity of its access methods.

8. References

Anick, P. G., Flynn, R. A., & Hanssen, D. R. (1991, Oct.). Addressing the requirements of a dynamic corporate textual information base. *Proc. 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Chicago, 163-172.

ATA/AIA Subcommittee 89-9C (1990, June). *CD-ROM Interchangeability Standard - SFQL: Structured Full-Text Query Language*, ATA Draft Standard 89-9C.SFQL2-R1-1990, version 2.0.

Burkowski, F. J. (1992). An algebra for hierarchically organized text-dominated databases. *Information Processing and Management*, Pergamon Press, New York, To appear.

Cooper, W. S. (1983). Exploiting the maximum entropy principle to increase retrieval effectiveness. *Journal of the American Society for Information Science*, 34(1), 31-39.

Croft, W. B. & Das, R. (1990, Sept.). Experiments with query acquisition and use in document retrieval systems. *Proc. 13th International ACM/SIGIR Conference on Research and Development in Information Retrieval*, Brussels, 349-368.

Date, C. J. (1990). *Relational database writings, 1985-1989,* Reading, Mass.: Addison-Wesley Publishing Co.

Fagan, J. (1987). *Experiments in automatic phrase indexing for document retrieval: A comparison of syntactic and non-syntactic methods.* Ph. D. Thesis, TR 87-868, Dept. of Computer Science, Cornell University.

Gonnet, G. H. & Tompa, F. W. (1987, Sept.). Mind your grammar: a new approach to modelling text. *Proc. 13th International Conference on Very Large Data Bases,* (VLDB87), Brighton, England, 339-346.

Gonnet, G. H. (1987). *Examples of PAT applied to the Oxford English Dictionary.* Centre for the New Oxford English Dictionary, Univ. of Waterloo.

Gyssens, M. J., Paredaens, J. & Van Gucht, D. (1989). A grammar-based approach towards unifying hierarchical data models. *Proc. ACM SIGMOD International Conference on Management of Data,* Portland, Oregon, 263-272.

Harman, D. & Candela, G. (1990). Retrieving records from a gigabyte of text on a minicomputer using statistical ranking. *Journal of the American Society for Information Science,* 41(8), 581-589.

Hoppe, H. U., Ammersbach, K., Lutes-Schaab, B. & Zinßmeister, G. (1990, Sept.). EXPRESS: An experimental interface for factual information retrieval. *Proc. 13th International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Brussels, 63-81.

International Standards Organization (1986, October). *Information Processing - Text and office systems - Standard Generalized Markup Language (SGML),* (ISO 8879), Geneva: ISO.

International Standards Organization (1988, March). *Information Processing - Text and office systems - Office Document Architecture (ODA) and Interchange Format, Part 1: Introduction and general principles* , (ISO 8613-1), Geneva: ISO.

McAlpine, G. & Ingwersen, P. (1989, June). Integrated information retrieval in a knowledge worker support system. *Proc. 12th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Cambridge, Mass., 48-57.

Meghini, C., Rabitti, F., & Thanos, C. (1991). Conceptual modeling of multimedia documents. *Computer,* 24(10), 23-30.

National Information Standards Organization (U.S.) (1989). *Information Retrieval Service and Protocol: American National Standard for Information Retrieval Service Definition and Protocol Specification for Library Applications,* Transaction Publishers, NewBrunswick, N.J.

Rau, L. F. & Jacobs, P. S. (1991, Oct.). Creating segmented databases from free text for text retrieval. *Proc. 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Chicago, 337-346.

Raymond, D. R. (1991). Reading source code. *Technical Report TR 74.070,* Centre for Advanced Studies, IBM Canada Ltd., Dept. 81/894, 895 Don Mills Road, North York, Ontario, M3C 1W3, Canada.

van Rijsbergen, C. J. (1979). *Information retrieval,* (Second Ed.), London: Butterworths.

Sacks-Davis, R., Wallis, P., & Wilkinson, R. (1990, Sept.). Using syntactic analysis in a document retrieval system that uses signature files. *Proc. 13th International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Brussels, 179-192.

Salton, G. & Buckley, C. (1990, June). An evaluation of text matching systems for text excerpts of varying scope. *Technical Report TR 90-1134,* Dept. of Comp. Sci., Cornell University, Ithaca, New York.

Salton, G. & Buckley, C. (1991, Oct.). Automatic text structuring and retrieval - Experiments in automatic encyclopedia searching. *Proc. 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Chicago, 21-30.

Tague, J., Salminen, A., & McClellen, C. (1991, Oct.). Complete formal model for information retrieval systems. *Proc. 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Chicago, 14-20.

Thompson, R. H., & Croft, W. B. (1989). Support for browsing in an intelligent text retrieval system. *Int. J. Man-Machine Studies,* (30), 639-668.

Wendlandt, E. B. & Driscoll, J. R. (1991, Oct.). Incorporating a semantic analysis into a document retrieval strategy. *Proc. 14th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval,* Chicago, 270-279.