Towards a Comparison of Structured and Unstructured Text Retrieval

Yves Marcoux and Christine Dufour
EBSI, Université de Montréal
marcoux@ere.umontreal.ca
dufourch@ere.umontreal.ca
http://tornade.ere.umontreal.ca/~marcoux/grds/

Structured text retrieval is the possibility of using the structure of documents (e.g., SGML tags) in search criteria, in addition to full-text operations (adjacency, etc.). Intuitively, using the structure can improve the performance of the retrieval, but for many reasons, it is difficult to compare structured and unstructured retrieval by real experimentation, and it has not been done yet. This article presents the methodology and preliminary results of a research project aimed at tackling the question of how structured retrieval compares to unstructured retrieval.

Le repérage textuel structuré est la possibilité d'utiliser la structure des documents (e.g., des balises SGML) dans les critères de recherche, en plus des opérations sur le texte intégral (adjacence, etc.). Intuitivement, l'utilisation de la structure peut améliorer le repérage, mais pour différentes raisons, il est difficile de comparer les repérages structuré et non structuré par réelle expérimentation, et cela n'a jamais été fait à ce jour. Le présent article présente la méthodologie et certains résultats préliminaires d'un projet de recherche ayant pour but d'étudier, à coût modique, comment les repérages structuré et non structuré se comparent.

Problem Setting

In a nutshell, research on structured documents is concerned with the boundary between information containers and contents. Traditional databases (most notably, relational) propose rigid containers (fields), highly predictable in shape, and thus easily treatable by simple syntactic rules. However, for textual data, the rigidity of the containers reduces their usefulness, resulting in semantic poverty. For example, if one wants to include full-text in a bibliographic-type database, one must place the whole text in a single field (or at best, in a fixed number of fields), making it impossible, for instance, to restrict a search to section titles or figure captions. The fact that some piece of text is located in the full-text field tells us little, if anything, more than the piece of text itself: the field has virtually no semantics attached to it. Thus, in effect, we end up in a situation similar to that at the other end of the structure spectrum: no structure at all, that is, an unstructured stream of natural language text. Unstructured text is somewhat treatable, but not by simple syntactic rules: it is very complex to extract meaning from it. Moreover, it is not easy to extract just as much semantics as would be conveyed by a database field; in order to extract any (correct) semantics at all from full-text, you must understand it fairly thoroughly.

The structured-document approach is to add inexpensive, field-like semantics to full-text. Thus, for instance, one would delimit all titles with the tags "<TITLE>" and "</TITLE>", so they would be easily identified (by simple syntactic means), and thus, could be attached the field-like semantics "TITLE". The key point, however, is that titles can be associated not only to the whole document, but to portions of it, such as sections, subsections, tables, etc. In other words, one does not have a single field "TITLE", but as many as necessary for any particular document. Moreover, these "fields" occur in context, that is, at a particular location in the document, and knowledge of that location enriches the semantics of the tag. (One could have multiple "TITLE" fields in a traditional "flat-file" database as well, by using a repeatable field; however, the titles would then be completely pulled out of context, and thus, all associated to the document as a whole, not to parts of the document.)

By allowing a rich semantics to be associated with each part of a document, the structured document approach encourages *data reuse.* For example: producing a procedures manual *and* a tools-to-procedures cross-reference listing, both from the same electronic document. Or: preparing the online help files *and* the printed documentation of a software product, both from the same electronic documents. Or: producing a history manual *and* a time-line chart, again from the same electronic document. As organizations discover that the use of semantic tagging (for example, SGML, XML, or HTML tags) in electronic documents facilitates information reuse, more and more published electronic documents include some form of explicit marking of their logical structure.

Can the explicit marking in structured documents be leveraged to improve the efficiency of information retrieval? It is very natural to imagine a positive answer to that question. First, it is easy to imagine specific situations in which restricting a search to, say, table captions, would certainly improve precision compared to full-text search. Second, one can consider searching to be a special form of data reuse; thus, if semantic tagging facilitates reuse, it should "facilitate" searching. But these mere reflections say nothing of real users in real search situations. The only valid way of establishing any benefits would be through user observation in real search situations.

However, comparing structured and unstructured retrieval is difficult, mainly because: (i) implemented approaches (e.g., software products) to structured retrieval use structural information in a fairly limited way; (ii) more powerful approaches, suggested in the literature (see Navarro and Baeza-Yates, 1995, for an excellent survey), have never been implemented beyond the stage of prototype; (iii) the usual efficiency measures of recall and precision are not directly applicable to structured retrieval. For reason (i), experiments based on implemented approaches would not give much information on true, powerful structured retrieval. For reason (ii), few corpora searchable with powerful tools exist, and searchers mastering these tools are scarce. Thus, experiments based on prototyped approaches, though possible, would probably be very difficult and costly to set-up. One possible research agenda would be to wait for the development—and adoption by users—of powerful commercial products, and then evaluate their efficiency. However, such products may never actually be developed, unless some indication can be given that they might improve retrieval efficiency, at least in certain circumstances.

What seems to be needed is a kind of evidence that would be intermediate between the more or less "obvious" intuitions as to how structured retrieval compares to unstructured full-text retrieval, and full-blown user-centred experiments. This is exactly the goal of the research project described in this article.

Overview of the Research Project

Goal of the Project

The main goal of the research project reported here was to provide some low-cost evidence about whether the "obvious" intuitions as to how structured retrieval compares to unstructured retrieval are right. In other words, we try to formalize and generalize these intuitions to a certain extent, and then verify them by heuristic simulation (that is, by humanly simulating search processes). Of course, human simulation is likely (if not certain) to introduce biases in the measurements. Still, this was deemed acceptable for our purposes, since all we wanted was rudimentary evidence.

Methodology

The research was conducted as follows: first, we defined efficiency measures (including precision and recall, but also new measures, such as query-preparation and results post-processing efforts) in a way applicable to both structured and unstructured retrieval. An effort was made to use objective indicators wherever possible, and to capture the search process in its whole.

Because, *a priori*, the benefits of structured retrieval, if any, could be influenced by many factors, we wanted to cover as broad a spectrum of search situations as possible. Thus, we identified 24 search *contexts*, a context being defined as the combination of a type of corpus, a type of user, and a type of information need. For each context, we identified two plausible information needs. For each need, we simulated the search process as it would be performed with both a structured and an unstructured tool. Finally, we applied our efficiency measures to each simulation. The simulations were all performed by the same research assistant, over a period of a few weeks.

The raw measures were collected in a relational database (Microsoft Access) and calculations of various indicators were performed in that environment. Graphics were produced from data exported to Excel.

We considered three types of corpora (book, collection of books, and collection of articles) and chose one corpus of each type. We used two types of users (novice and expert) and four types of information needs (familiarization, known-item, specific, and general) for a total of 24 contexts and 48 needs. As a model of unstructured retrieval, we used a slightly modified set of common full-text retrieval commands (truncation, distance, etc.). As a model of powerful structured retrieval, we used the model of Marcoux and Sévigny (1997).

Structure of the Article

In the next four sections, the various components of our search contexts will be explained in more detail: the types of corpora, the retrieval languages, the types of information needs, and the types of users. The two following sections introduce the simulation process and a few of the indicators used to measure retrieval efficiency and user effort. Next, we present preliminary results. Finally, we conclude and point out possible future work. The article assumes a basic knowledge of SGML.

The Corpora

We used three SGML corpora exhibiting different types of "macrostructure", or ways of grouping information: (1) a book on a given topic, (2) a collection of books presenting different aspects of a topic, and (3) a collection of short articles on a given topic. Apart from their macrostructure, the corpora were chosen mainly for their free availability, and because they were all readable with a viewer allowing full-text (unstructured) retrieval as well as some basic form of structured retrieval (either Dynatext or Panorama Pro). Thus, it was possible to execute unstructured searches almost directly, and to simulate powerful structured retrieval with minimal difficulty.

The corpora used in this project were made up of SGML documents. One of them was a single SGML document, whereas the two others comprised many separate SGML documents. Because the structured retrieval language (see next section) works on single SGML documents (called *document-bases*), we conceptually added a top element (tagged <SET>) to the two composite corpora.

Final Draft of the DSSSL Standard

We chose the final draft of the *Document Style Semantics and Specification Language* (DSSSL) standard as a corpus of type "book". This book was available in two different formats: SGML (readable with Panorama Pro) and Dynatext format. Both the structured and unstructured searches were performed with Dynatext; however, for two of the information needs, the four corresponding searches were performed with Panorama Pro.

Netware 4.02 Manuals (French Version)

Eight of the original twelve Novell's manuals were used to represent the "collection of books" corpus type. All the manuals have the same structure, i.e., the same SGML Document Type Definition (DTD). The manuals were provided free of charge by Novell Corporation.

The books are available in two formats: SGML and Dynatext. Dynatext and Panorama Pro were used to perform both structured and unstructured searches. One of the manuals in Dynatext format was corrupted, so we used the SGML version with Panorama Pro to perform the searches on it.

SGML World Tour

In 1994, Softquad produced a CD-ROM containing several texts related to SGML entitled "SGML World Tour." It is divided into several sections: "Starting to understand SGML", "Frequently Asked Questions", "Starting to Use SGML", etc. We chose the section "Starting to Use SGML", that contains nineteen introductory articles, to represent the "collection of articles" corpus type. The main difference between this corpus and the collection of books corpus is that all the articles do not have the same structure: they are based on four different DTDs.

Although this corpus came with Softquad Explorer on the CD-ROM, we used the SGML forms of Panorama Pro to perform searches, because the querying capabilities of Softquad Explorer were not appropriate for our purpose.

The Retrieval Languages

Two different retrieval (or query) languages are considered in this project: an *unstructured* (or "structure-unaware") language, and a *structured* (or "structure-aware") one. Roughly speaking, the difference resides in the possibility of using SGML tags within the search criteria of the queries.

Structured Retrieval Language

We use the structured query language defined by Marcoux and Sévigny (1997). Here is a brief overview of this language.

Normal full-text retrieval operations form what is called *word expressions*, which are always enclosed in square brackets [ ]. Word expressions can (only) match the textual content of an SGML element (no tags). The word expression "[$]" matches any textual content. Expressions matching generic IDs, called *genid expressions*, can also be specified. They are of the form "TI" or "TI*" or "TI | TITLE" or "S* ∩ *SECTION", where "*" represents truncation, " | " union (more or less the Boolean OR), and "∩" intersection (more or less the Boolean AND). The last example would match the generic IDs "SECTION", "SUBSECTION", etc., but not "INTERSECTION". The genid expression "*any*" matches any generic ID. Word expressions and genid expressions can be combined to form *ordinary expressions*, which are the basis of search expressions. Here are the (recursive) rules for building ordinary expressions: any word expression is an ordinary expression; also, if a is a genid expression, and b and g are ordinary expressions, then "$<\alpha>\beta</>$", "(β, γ)", "$(\beta \mid \gamma)$", "$(\beta \cap \gamma)$", "$(\beta+)$", "$(\beta*)$", "$(\beta?)$", and "$(\neg\beta)$" are ordinary expressions. The expression "$<\alpha>\beta</>$" matches any segment "$<x>y</>$" of the document such that x matches α and y matches β. The operators ",", " | ", " + ", "*", and "?" have the same semantics as in SGML; "∩" represents intersection (more or less the Boolean AND), and "¬" represents Boolean negation (or complementation). There are restrictions on the use of "¬". The comma (sequence operator) and parentheses are often omitted. A *query* is a Generalized *Context-Free Grammar* (see Hopcroft and Ullman, 1979), that associates an ordinary expression to certain symbols (*non-terminals*), in such a way that non-terminals can be used (possibly recursively) in the ordinary expression associated with other non-terminals (or the same). A distinguished non-terminal, *S*, is called the *start-symbol*, and it must match the whole SGML document for the query to return a non-empty result. The result of a query is a (possibly empty) set of sequences of consecutive SGML elements. The sequences returned in the result correspond to expressions (or sub-expressions) that were *underlined* by the user in the query.

Here is an example of a query:

$$
\begin{gathered}
S \rightarrow (\omega\ \mathrm{R}\ \omega) \mid (<\mathit{any}>S</>) \\
\omega \rightarrow [\$] \mid (<\mathit{any}>\omega</>) \mid (\omega^{*}) \\
\mathrm{R} \rightarrow (<\mathrm{SEC}>\ \omega\ <\mathrm{TI}>[\mathrm{art}]</>\ \omega\ </>)\ (<\mathrm{SEC}>\ \omega\ \underline{<\mathrm{TI}>}\omega</ \\
>\ \omega\ </>)
\end{gathered}
$$

This query would return the titles of sections that immediately follow any section having the word "art" it its title. Observe that the non-terminal w matches any (well-formed) document segment whatsoever. In the remainder of this article, we assume *S* and ω are defined as above, unless otherwise stated.

The operations allowed within word expressions are:

Operation	**Notation**
contains word	word
contains nothing else but string	"string"
Boolean OR	|
Boolean NOT	¬
one-character mask	?
word truncation (anywhere)	*
proximity in the order specified, *n* words	*n*P
proximity in any order, *n* words	*n*D
contains anything (matches any text content)	$
intersection (Boolean AND)	∩

Note that the query language does not have a notion of SGML attributes; however, attributes can be considered to be sub-elements, occurring as last children of the element to which they apply.

Marcoux and Sévigny (1997) mention an extension of the query language allowing some formatting or hyperlink oriented tags to be ignored. This would allow the word expression "[impres* 3D art]" to match "Other art forms in the <BOLD> impressionist </> period", even though "impressionist" is actually in a sub-element. We assume that a pre-determined set of tags has been identified to be ignored.

Unstructured Retrieval Language

The original corpora were structured; we considered their unstructured versions to simply be the corpora without SGML tags. However, all text normally added by way of style sheets (for example, affixes, such as the word "Chapter" preceding a chapter sequence number at the beginning of a chapter) was considered to be part of the unstructured versions.

The unstructured query language is strongly based on common full-text retrieval languages. The operations allowed are:

Operation	**Notation**
single word	word
adjacency	[space] (implicit)
Boolean OR	|
Boolean AND	&
one-character mask	?
word truncation (anywhere)	*
proximity in the order specified, *n* words	*n*P
proximity in any order, *n* words	*n*D
non-proximity in the order specified, *n* words	¬*n*P
non-proximity in any order, *n* words	¬*n*D

We can combine operators and use parentheses to specify priority. Note that the Boolean negation by itself is not included, because it makes sense only in the context of field-structured databases. Suppose we searched for "¬art" and the corpus did not contain the word "art", then the query would be successful, but the system would have no hit to display. It should also be noted that Boolean operation AND was included, but it has very little use, since it applies to the whole corpus.

Search Results Presentation

Since our measures include indicators aiming at capturing the search process as a whole, such as the effort to extract the answer from what is returned by the system, we need to assume something about how the search results are presented to the user.

The basic scheme is that a query returns a (possibly null) number of results, each result being a continuous document segment. In the unstructured case, the document segments returned are defined to be 100 characters in length, and centered on a "hit". A hit is defined to be an occurrence of one of the search terms from the query, in a context that matches the query. This agrees with the usual notion of a hit in full-text search environments. Thus, there are as many results to a query as the number of hits it generates. In the structured case, recall that a query returns a set of sequences of consecutive SGML elements. Now, a sequence of consecutive SGML elements always corresponds to a continuous document segment. Thus, we define the results returned by a query to be the document segments corresponding to the sequences of SGML elements returned by the query.

We assume, in both cases, that the results are presented sequentially to the user (in their order of appearance in the document), in a list that clearly shows the boundaries between results. We also assume there are hyperlinks allowing the user to jump from the result list to the actual document and that, in the actual document, the segments corresponding to results appear highlighted. We assume the presence of navigational buttons to directly reach the previous or the next result and of a density scale to facilitate the navigation. We also assume that there is a visual indication if a given result has already been consulted (for example, a different color), and that, within each result (in the result list or in the actual document), the search terms from the query (if any) are shown with special presentation attributes (e.g., in reverse video).

The Information Needs

Users have different types of information needs. Following Meadow (1992, 243-244), we categorize them under four types: exploration of the document base (browsing), specific-information search, known-item search, and general-information search.

Exploration of the Document Base (browsing)

This is a first exploratory contact with a document base. The user tries to build his/her own representation of the document base by getting acquainted with the kind of information it contains. In structured retrieval, the structural elements can be used to meet this kind of need by, for example, making a query that builds a table of content or extracts the introduction section. The only way to explore the document base in unstructured retrieval is to browse in a more or less linear manner. The user would probably pay more attention to the beginning of the document, where s/he would likely find an introductory part and at the end where s/he would likely find a conclusion.

Specific-Information Needs

The user needs a specific piece of information such as, for example, the occurring date of an event. The characteristics of specific information is that the user will know when s/he has found it. In the case of a date, for instance, the user will recognize that s/he has obtained an answer when presented a piece of information of the form "Such and such an event occurred on Jan-1-1970".

Known-Item Needs

The user searches for complementary information on a known-item such as the first name of a person when the last name is known. We include here searches for information that the user has seen before in the document and remembers partially.

General-Information Needs

The user needs information on a topic in general; for example, information on computer interface design or a method to resolve differential equations.

The Users

We define two types of users: novice and expert. The differences between them reside in their knowledge of the content and of the structure of the document base. We assume that the novice has no knowledge of the structure, but has the required background to understand the document base content. The expert is familiar with both the content and the structure, from prior utilization of an equivalent of the document base in another format, such as a printed version. We assume that the query languages and the systems used to query the corpora are already known.

The novice was considered at two stages of utilization of the corpus:

1. As s/he does not know the document base, the first stage of utilization consists of an exploration. At this stage, s/he can have only exploratory needs or specific-information needs.

2. In the second stage, s/he tries to resolve deeper information needs. Specific-information needs, general-information needs, and known-item searches can be encountered here.

We assumed that experts, by their previously acquired knowledge of the document base, do not have an exploratory stage. They have only specific-information needs, know-item needs, and general-information needs.

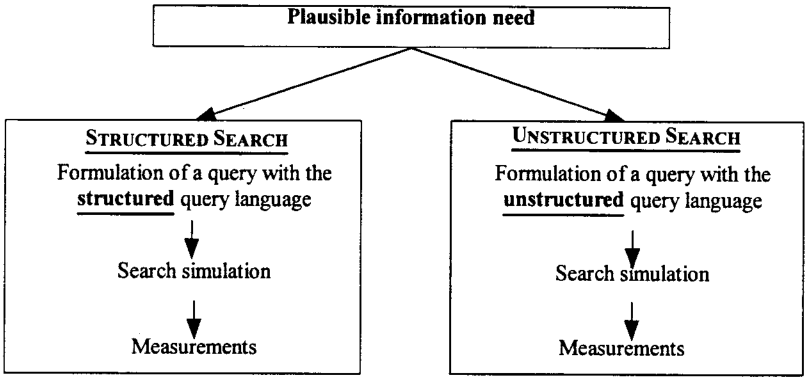
The Simulations

Proceeding by simulation not only was more economical than experimentation, but made the research possible, because the structured language has never been completely implemented so far.

Searches (both structured and unstructured) were simulated using the system on which each corpus was delivered (either Dynatext or Softquad Explorer) or Panorama Pro. The tightest possible search was performed using the capabilities of the query language of the system and, when necessary, additional manual operations were performed to exactly establish the results of the simulated query.

For each corpus, each user type, and each type of information needs applying to that user type, we selected two plausible information needs, and performed the following steps for each need:

Figure 1. Simulation Steps


Plausible information need
STRUCTURED SEARCH
Formulation of a query with the structured query language
Search simulation
Measurements
UNSTRUCTURED SEARCH
Formulation of a query with the unstructured query language
Search simulation
Measurements

The "measurements" steps consist in evaluating a number of indicators, some of which will be introduced in the next section. The same research assistant performed all simulations, taking great care to be as consistent as possible. Because of the objective nature of many of the indicators used, biases are introduced by the simulation only at the following points: selection of the information needs; formulation of the queries; identification of the section(s) of the document-base relevant to the information need.

We consider each need independently from the others (except that the novice at the second stage is considered to have gone through the first stage); the knowledge presumably acquired by solving a need is completely ignored for solving other needs.

A total of 96 simulations were conducted, corresponding to 48 information needs (each need giving rise to two simulations). The following table breaks down the 48 information needs used.

Figure 2. Breakdown of Information Needs

USER TYPE OF NEEDS	STAGE	INFORMATION NEEDS TYPE	NUMBER
NOVICE	stage 1	exploratory needs	2
		specific-information needs	2
	stage 2	specific-information needs	2
		known-item needs	2
		general-information needs	2
EXPERT		specific-information needs	2
		known-item needs	2
		general-information needs	2
			16
			x 3 corpora =
			48 information needs

The Indicators

We defined many indicators for measuring query-preparation effort, post-processing effort, and retrieval efficiency. Due to space limitation, we shall only present a few of them. We present indicators in three groups: those aiming at measuring the query-preparation effort, those aiming at measuring the efficiency of retrieval, and those aiming at measuring the result post-processing effort.

In a real experimentation, the user effort would probably be measured by elapsed-time and/or user-judgement indicators. In our case (simulation), such an approach is not applicable. Instead, we define indicators that aim at capturing the complexity of the tasks involved in an objective manner. Thus, the two "user-effort" groups of indicators include only objective indicators.

We shall use the following example, taken from the actual information needs used in the project:

Example of information need
Corpus: Netware Manuals
User type: Expert
Information need category: Specific-information need
Information need: What is the procedure to configure TCP/IP?

Structured query:
```
R → <title>ω[configur* 5D TCP/IP]ω</> T
T → ω (<procedure>ω</> | <any>T</>) ω
```
Unstructured query: `configur* 5D TCP/IP`

Indicators to Measure Query-Preparation Effort

Intuitively, the query-preparation effort is influenced by the complexity of the query itself in both the structured and unstructured cases. In the structured case, we must also take into account the intrinsic complexity of the document-base structure (because that structure must be understood before it can be used in formulating the query), as well as the depth of the structure exploration required prior to query formulation.

Query Complexity

Lexical Complexity

We measure the lexical complexity of a query by counting the number of operators and search terms in the query. It is straightforward to evaluate this indicator for unstructured queries. For structured queries, we count search terms occurring in genid- as well as word-expressions. Underlining is counted as 1, as are "$\neg$", "|", "$\cap$", "+", "*", "?", "\$", and the construction "$<\alpha>\beta</>$". Omitted commas and the genid expression *any* are not counted.

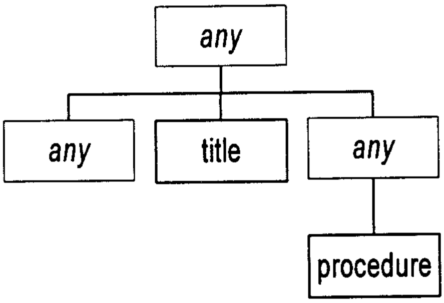
It can be verified that for the unstructured query of our example, this indicator is 4 (2 terms and 2 operators). For the structured query, it is 12 (4 terms and 8 operators).

Nesting

This indicator, defined only for structured queries, counts the maximum level of nesting of "$<\alpha>\beta</>$" constructions in the query. Since recursions in the non-terminals have their own indicator, we count them here as one level of nesting (this is as if we developed one level of the recursion). The nesting in the definitions of S and ω (shown in Section 4.1) are counted.

A hierarchical representation of the query can facilitate the evaluation of this indicator. Here is the structured query of our example represented as a tree, showing that the level of nesting is 3. The top-level box corresponds to the *any* occurring in the definition of S.

Figure 3. Tree Representation of the Structured Query Example



Number of Recursions

This is another indicator defined only for structured queries. The recursions contained in a query contribute to its complexity. Thus, we take as another indicator of the complexity of the query the number of non-terminals with a recursive definition. The recursion in the definition of S is counted, but not the one in the definition of ω (ω is considered "built" into the system, which was the case in the prototype developed by Marcoux and Sévigny (1996) for their model). In the example structured query, the number of recursions is 2.

Other Indicators

As mentioned previously, we also have indicators to estimate the intrinsic complexity of the structure of the document-base, as well as the depth of the structure-exploration required prior to formulating the query. Due to space limitation, we omit the definition of these indicators.

Indicators to Measure Retrieval Efficiency

To estimate the retrieval efficiency, we use the well-known indicators **recall** and **precision**. We must redefine them to fit both structured and unstructured retrieval. Our definitions are based on the following hypothesis:

> *For any given information need, an expert (or many experts) can identify a number of segments of the document base such that, if the user consults those segments and only those segments, then s/he can fulfill his/her information need completely, and without undue effort.*

We call those segments the **response elements** corresponding to the information need. We now state our definitions of precision and recall in terms of the following scenario: Imagine the expert underlines the response elements in the document-base, and the retrieval system underlines the segments corresponding to the results returned by the query. Then, we define:

precision = (# of characters underlined twice) / (# of characters underlined by the system)
recall = (# of characters underlined twice) / (# of characters underlined by the expert)

As usual, if a denominator is null, we define the corresponding measure to be 1. Markup characters were not counted in the structured case, although it could be argued they should.

For our example, unstructured recall and precision were respectively 0,027 and 0,009, while both measures were 1 in the structured case.

Indicators to Measure Post-Processing Effort

We now briefly sketch two of the indicators aimed at measuring the result post-processing effort on the part of the user.

Distance Between Responses and Results
Sometimes, the system returns only part of the response elements. In that case, the user has to read above and under the results to determine where the response elements begin and where they end. To approximate this effort, we measure the distance between the beginning of a response element and the beginning of a result, and between the end of a result and the end of a response element (there are many different cases, that must be treated differently).

First Response Element Position
This indicator counts the number of results a user has to consult before s/he encounters a first response element. This is specially meaningful for specific-information needs, since the user is likely to stop consulting results immediately after the first response element.

Preliminary Results

To illustrate the results obtained, we present a few comparative charts for recall, precision, and lexical complexity of the query.

Figure 4. Precision Comparison, Novice, Stage 2

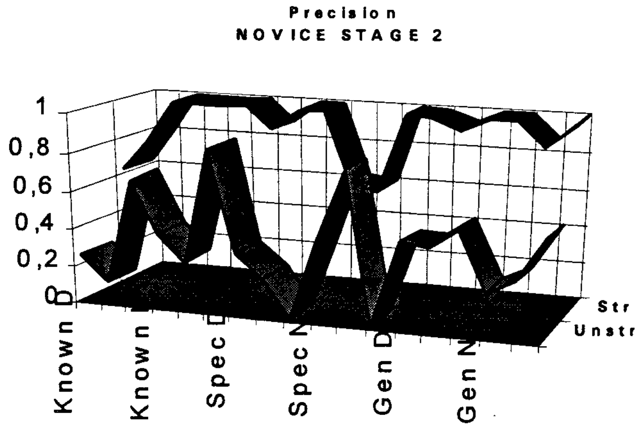


Figure 5. Recall Comparison, Netware

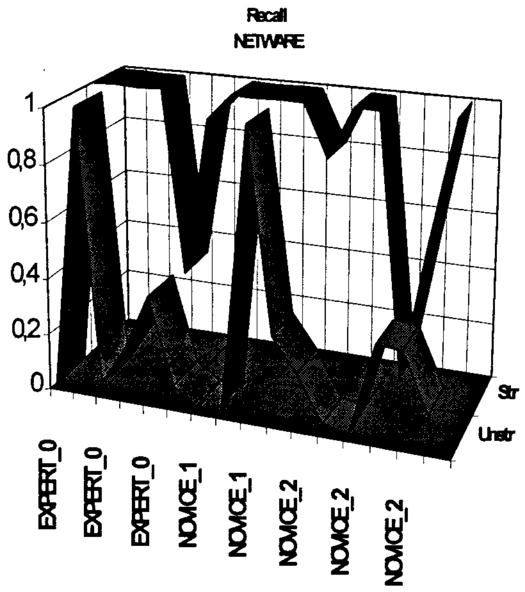
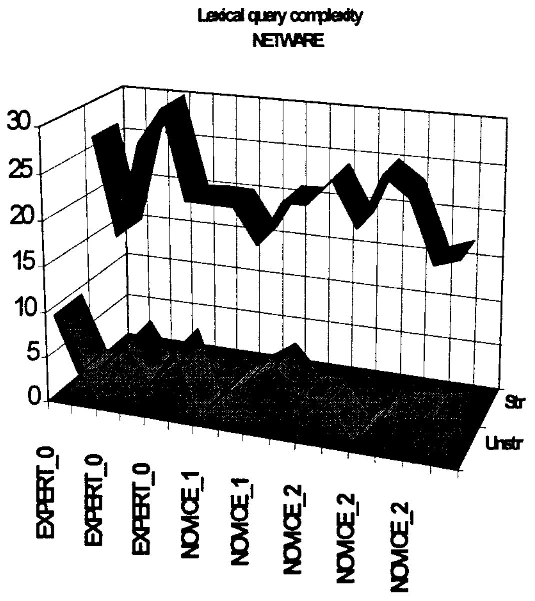


Figure 6. Lexical Query Complexity, Netware



These results, as well as other ones, will be commented upon at the conference.

Conclusion and Future Work

Though our results cannot be interpreted in terms of real search situations (because they were obtained by human simulation) and cannot be generalized, they do, nevertheless, support the intuition that structured retrieval gives better precision than unstructured retrieval. A surprising aspect observed so far is that recall does not suffer much from structured retrieval. A not too surprising aspect is that the query preparation effort seems to be much higher in the structured case. At the time of this writing, we have not yet analyzed the result post-processing indicators enough to draw any conclusion on that aspect.

Further analysis of the data gathered so far, as well as refinements of our various indicators might give additional insight. The ultimate step would be to perform a true comparison of structured and unstructured retrieval by experimentation. This would require at least the development of a fully functional prototype incorporating the structured retrieval language, which is not planned at the present time. An intermediate step would be to identify users who need to consult corpora available in SGML or XML on a regular basis, train them to the use of the structured query language, ask them to formulate structured and unstructured queries for their real information needs, and go on simulating the searches as in the present project. This would have the advantage of allowing the query-preparation effort to be measured by more "user-centered" indicators, such as elapsed-time and/or user judgement, rather than objective indicators. However, result post-processing effort would still be measured by objective indicators.

Acknowledgments

This research benefited from the financial support of the Social Sciences and Humanities Research Council of Canada.

The authors wish to thank Novell Corporation for providing free of charge some of the SGML documents used in this project.

References

Hopcroft, John E., and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation.* Don Mills, Ontario: Addison-Wesley, Inc., (Addison-Wesley Series in Computer Science).

Marcoux, Yves, and Martin Sévigny. 1996. "Querying hierarchically structured text with generalized context-free grammars." Nineteenth annual international ACM SIGIR conference on research and development in information retrieval. Software demonstration—abstract in the Conference Proceedings.

Marcoux, Yves, and Martin Sévigny. 1997. "Querying hierarchical text and acyclic hypertext with generalized context-free grammars." *RIAO 1997 Conference Proceedings*, Paris: Centre de hautes études internationales d'informatique documentaire, pp. 546-561.

Meadow, Charles T. *Text Information Retrieval Systems.* 1992. California: Academic Press, Inc., (Library and Information Science Series).

Navarro, G., and R. Baeza-Yates. 1995. "A language for queries on structure and contents of textual databases." *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 93-101.