Susan E. Teague Rector. Accessing Information Based on a Combination of Document Structure and Content: Exploiting XML tags in indexing and searching to enhance content retrieval of online document-centric XML encoded texts. A Master's Paper for the M.S. in I.S. degree. April, 2004. 64 pages. Advisor: Stephanie W. Haas

This study explores the challenges of using traditional information retrieval methods to retrieve document-centric XML encoded text. It demonstrates how coupling structure and content in query and index formulation improves retrieval performance. Native XML database (NXD) and search engine technologies were evaluated in a baseline experiment, and in a second test after alterations were made to their respective indexes. Documents were retrieved for simple and complex forms of 30 XPath and keyword queries from a corpus of 95 XML/TEI encoded texts. Overall results indicated that query augmentation using document structure improves retrieval performance. Complex queries submitted to the NXD produced the most satisfying results, with an average precision of 93.3% and an average recall of 86.3%. Performance improvements were also achieved using complex, structured queries and indexes in the search engine. Study findings suggest that effective XML retrieval models might result from a combination of unstructured and structured retrieval techniques.

Headings:

Information Retrieval

XML search & retrieval

Semistructured Data Indexing

Full Text Searching

ACCESSING INFORMATION BASED ON A COMBINATION OF DOCUMENT STRUCTURE & CONTENT

Exploiting XML tags in indexing and searching to enhance content retrieval of online document-centric XML encoded texts

by
Susan E. Teague Rector

A Master's paper submitted to the faculty
of the School of Information and Library Science
of the University of North Carolina at Chapel Hill
in partial fulfillment of the requirements
for the degree of Master of Science in
Information Science.

Chapel Hill, North Carolina

April 2004

Approved by

____

Stephanie W. Haas

TABLE OF CONTENTS

1.0 Introduction

The increasing popularity and growth of internet technologies over the past 10 years has brought with it myriad challenges to document management. There are an ever-growing number of proposals in information technology (IT) publications and information science (IS) literature to address searching, managing and organizing the wealth of information now available online. One area receiving considerable attention from the information retrieval (IR), database and search engine communities is the retrieval of Extensible Markup Language (XML) encoded documents. Significant amounts of research and development have gone into the creation of document repositories, IR models and query languages suitable to the effective and efficient retrieval of XML documents.

Underlying this surge of research and development is a growing debate about which methodologies best fit XML retrieval. Will classical IR models or will conventional relational database techniques work to solve XML retrieval challenges? What is becoming apparent in all technology communities is that neither of these approaches fully and adequately addresses the retrieval of XML encoded documents. Rather, many are suggesting that the most practical models for XML retrieval might be found in systems and query languages that, at the same time, exploit the rich semantic features of XML markup and provide full-text searching capabilities. Could prior knowledge about a document’s structure inform the search process?

Where research activity is most visible is in the IR community where researchers are investigating the possibility of “hybrid” structured and content querying techniques to provide more relevant search results than keyword based approaches traditionally used in document databases. Could knowledge of XML document paths lead users to relevant information more quickly and easily? Although many studies have approached these questions from various angles, there is not yet one definitive answer. And, while XML retrieval is an emergent technology area, it remains one that is still in its infancy. More research is required to assess the strength of systems that query document collections using both keyword and structured data techniques. More research is needed to determine if the languages used to query XML are accessible to developers as well as to end-users.

The present study works to address some of these challenges surrounding XML retrieval. This research stems from the discovery that at the time of writing, there were few studies focusing on the performance aspects of structured path-based XML retrieval compared with keyword based searching in online XML document collections. This study explores the possibility of hybrid systems that retrieve information based on both document structure and content. Using open source native XML database solutions, search engine technology and novel XML query languages like XPath (Clarke & DeRose, 1999), it seeks to answer the question of whether exploiting XML tags in XPath queries improves and enhances precision and recall over traditional full-text retrieval, keyword based techniques.

2.0 Background and Related Research

2.1 Types of XML Documents

A language “originally designed to meet the challenges of large-scale electronic publishing,” XML is now used to model myriad types of information in various technology environments (World Wide Web Consortium Schools [W3C Schools], n.d., “*On SGML and XML*,” para.1). It has become the language of choice among such seemingly disparate communities as financial businesses, publishing organizations and digital libraries. Along with the increasing use of XML, two terms have emerged in the literature to describe XML documents: *data-centric* and *document-centric* (Bourret, 2003). Today, businesses commonly use data-centric XML to communicate information between their organizations. Bourret (2003) remarks that typically these ephemeral, data-centric documents are kept only until the data has been received, mapped and entered into a relational database management system (RDBMS). In these data transport schemes, the data itself is what is most important; the “document entity” is inconsequential once the data has been mapped and stored (Bourret, 2003).

Preserving an encoded document in its original format is of great importance in electronic publishing, however. Publishers of electronic text have long realized the importance of well-defined, semantically rich document encoding languages like XML and its parent SGML. In a digital library for example, documents encoded in XML become part of a library’s permanent collection, and are thought of as the “electronic editions” of their physical counterparts if any exist. These electronic editions are often termed “semi-structured documents” because of the combination of structural elements such as the work’s metadata, and the unstructured sections of the document that correspond to large portions of text. Today, many researchers refer to these often large

XML encoded texts as document-centric XML. Document-centric XML takes advantage of the language's inherent ability to model "structural, presentational, and semantic information alongside content" (W3C Schools, n.d., "*On SGML and XML*", para. 1).

2.2 Predominant XML Query Languages

The method by which document-centric documents are accessed in many XML retrieval systems is either using XML Path Language (XPath) or XML Query Language (XQuery), both standards supported by the World Wide Web Consortium (W3C). XPath is "a language for addressing parts of an XML document" (Clark & DeRose, 1999, Section 1). It is considered to be a simple language to learn, as queries are formed using the hierarchical paths of the XML document. The language allows developers and users to specify "location paths" to nodes in the XML document (Harold & Means, 2002, p. 157). These paths are "built out of successive location steps…" where "each location step is evaluated relative to a particular node in the document called the context node" (p.157). This easy method of traversing a document makes XPath an attractive and easily implementable language in many XML retrieval systems today.

While some XML retrieval systems are moving toward integrating XQuery into their querying schemes, XPath remains the predominant language for querying XML in many open source software applications. Because XQuery will eventually subsume but not replace XPath, this study was conducted using the XPath standard, as it is a suitable language to test structure and content-based retrieval from XML document collections.

2.3 Major Movements in XML Retrieval Research

Although many XML retrieval systems still rely on data-centric techniques of mapping document structure to relational database tables, there is a growing movement in IR, database and search engine communities to research and develop hybrid systems to handle document-centric XML. Researchers in all communities are recognizing that while relational models may fit data-centric documents, they do not address document-centric information in a suitable manner. The studies covered here focus primarily on three of the most active areas in document-centric XML retrieval research: the IR communities' perspective on XML retrieval; database vendor applications and open source software solutions for XML retrieval; and the search engine communities' perspective on XML and the web.

2.3.1 XML Retrieval & Classical IR Techniques

Historically, IR researchers have focused their energies on optimizing content-based retrieval methodologies to effectively find relevant information in document databases. There is a rich history of IR techniques to store, index and extract information based document keywords (e.g. Baeza-Yates & Navarro, 1999; Chowdhury, 1999; Salton & McGill, 1983; van Rijsbergen, 1979). Content-based retrieval typically incorporates vector space, probabilistic or Bayesian models that refer to a unit of retrieval as an entire document entity (Baeza-Yates & Navarro, 1996). Baeza-Yates and Navarro (1996) write that content-based retrieval in document databases poses different IR challenges from those found in relational systems, where the data is mostly structured and the concept of a "document" is irrelevant for the most part (p. 68). Baeza-Yates and Navarro comment that in a document database, documents are normally indexed on their content and meaning, keywords are assigned as indexes or access points to the document, and the structure in a document is often ignored in both indexing and searching.

Currently, IR researchers are finding classical content-based retrieval schemes to be useful in the exploration of XML retrieval models. Many believe that these methods could be extended to enhance the information discovery process by accounting for both the structure (i.e., the metadata) and large text sections of XML documents. Baeza-Yates and Navarro (1996) argue that the integration of conventional content-based retrieval methods and relational database techniques could help end-users of information systems answer inexact or "fuzzy" types of IR questions (p. 68). They write that traditional IR systems "have allowed [users] to search their contents (words, phrases, etc.) or their structure (e.g. by navigating through a table of contents), but not both at the same time" (p. 67). Systems are beginning to emerge, they note, that enable the mixing of content and structural based queries (p.67). Combining these two methodologies could result in systems capable of answering questions such as "find all instances of the word *debt* that appear near the beginning of Chapter 1 in Charles Dickens' *Great Expectations*" (p. 68).

Like Baeza-Yates and Navarro (1996), Luk et al. (2002) believe that systems combining structured and content retrieval are inevitable as the number of XML documents increases exponentially on the web. They write that "XML holds the promise that searching can be done more precisely because structural, self-describing information and meta-data (e.g. RDF) is available to allow for context-based and/or category-based search" (p. 415). Their exhaustive survey details the major areas of XML IR related research, namely: effective indexing and searching methods for XML, XML query languages and XML retrieval models (p. 415). Despite the proliferation of XML retrieval research and development, Luk et al. remark that a practical retrieval model to handle structure and content-based documents is not yet realized; this is an exciting area for research (p. 415).

In order to properly assess XML retrieval technology, organizations are forming such as the Initiative for the Evaluation of XML Retrieval (INEX) supported by the IEEE Computer Society and DELOS, Network of Excellence on Digital Libraries. INEX is primarily devoted to providing international researchers with test data and benchmarks to determine the feasibility of "content-based XML retrieval." According to Fuhr, Govert, Kazai & Lalmas (2002), INEX seeks to promote research methods for document-centric XML. The group's query formulation methods, data, and benchmarks have been incorporated into many of the studies found in the literature and serve as the basis for some of the testing methods used in the present study.

Effective retrieval of XML is also a topic of much discussion in recent ACM SIGIR papers. At 2000 and 2002 workshops, several researchers (e.g. Baeza-Yates, Fuhr & Maarek, 2002; Carmel, Maarek, & Soffer, 2001) showed growing interest in discovering suitable models for XML retrieval. In 2000, they addressed how "integrating IR and XML search techniques will enable more sophisticated search on the structure as well as the content of these [XML] documents" (Carmel et al., 2001, p. 62). Some of the major challenges participants outlined align with other research groups, notably: discovering indexing techniques to take advantage of XML structure; searching XML "both on content and structure;" uncovering how XML structure can enable better context-based searching; and, how to exploit "database indexing techniques in an IR framework" (p. 62).

The main focus of the 2002 SIGIR conference was to “continue the effort of applying an “IR approach” to XML retrieval, and to investigate how the IR community could have more impact on organizations like the W3C…” (Baeza-Yates, Fuhr & Maarek, 2002, para. 2). A common goal for all participants was to provide a definition of XML retrieval apart “from pure data exchange (in business-to-business applications) to actual information exchange (in end-users facing and Knowledge Management applications)” (para. 2).

Much of the literature resulting from the SIGIR workshops concentrates on the central question of whether classical IR probabilistic or vector space content retrieval techniques are applicable to XML IR. Wolff, Florke & Cremers (2000) tested this idea by extending the probabilistic IR model, as they felt it “lacks the ability to handle structural information” in XML documents (p. 141). Like other researchers, Wolff et al. assert that “knowledge about the structure of documents is an additional resource that should be exploited during retrieval since the semantics of the different textual objects can be used to specify an information need more precisely” (p. 141). Using their own system, XPRES, and documents from the OHSUMED collection, they formulated two series of tests on documents, indexing with one entry point versus multiple entry points. Wolff et al. found that “a significant improvement of precision for lower recall values is achieved” using the collection with multiple indexes. Their findings indicated “that the explicit distinction of structural elements can improve the retrieval effectiveness” (p. 149). Moreover, their results suggested “that meaningful structural information can and should be utilized in exploiting document collections” (p. 149).

Myaeng, Jang, Kim & Zhoo (1998) also introduce a traditional IR based model for querying document-centric encoded documents similar to that of Wolff et al. (p. 139). They examined whether Bayesian and inference network models are suitable to semi-structured/document-centric retrieval. Running 32 queries on “a collection of 1000 documents (about 40MB)” from the TREC collection, Myaeng et al. discovered “that the use of structures in SGML documents indeed improves the retrieval effectiveness, especially precision” (p. 144). Their study demonstrated that the “use of structural information embedded in the SGML documents can actually improve the effectiveness of document retrieval, in comparison with the case where no such information is used” (p. 144).

A key aspect of effectively retrieving relevant content is having efficient indexing structures. Alongside testing conventional IR models, researchers are also examining indexing schemes in relation to XML retrieval. Kotsakis (2002) addressed this issue by looking at which indexing techniques might apply to document-centric XML. He argues that the movement in the database community has been largely to search XML using keyword based data-centric approaches. What is needed, he remarks, is an approach to querying semi-structured documents with IR techniques (p. 665). Using traditional IR inverted file structures, Kotsakis ran 100 queries on the “Cystic Fibrosis (CF) document collection,” a 6MB XML collection containing 1239 documents (p. 665). His test results suggest “that indexing the tags,” in XML data, “might be an effective approach to capturing document structure” in XML retrieval (p. 666). The 2002 Kotsakis study furthermore advances the idea that an inverted file structure is a practical solution to indexing XML documents (p. 666).

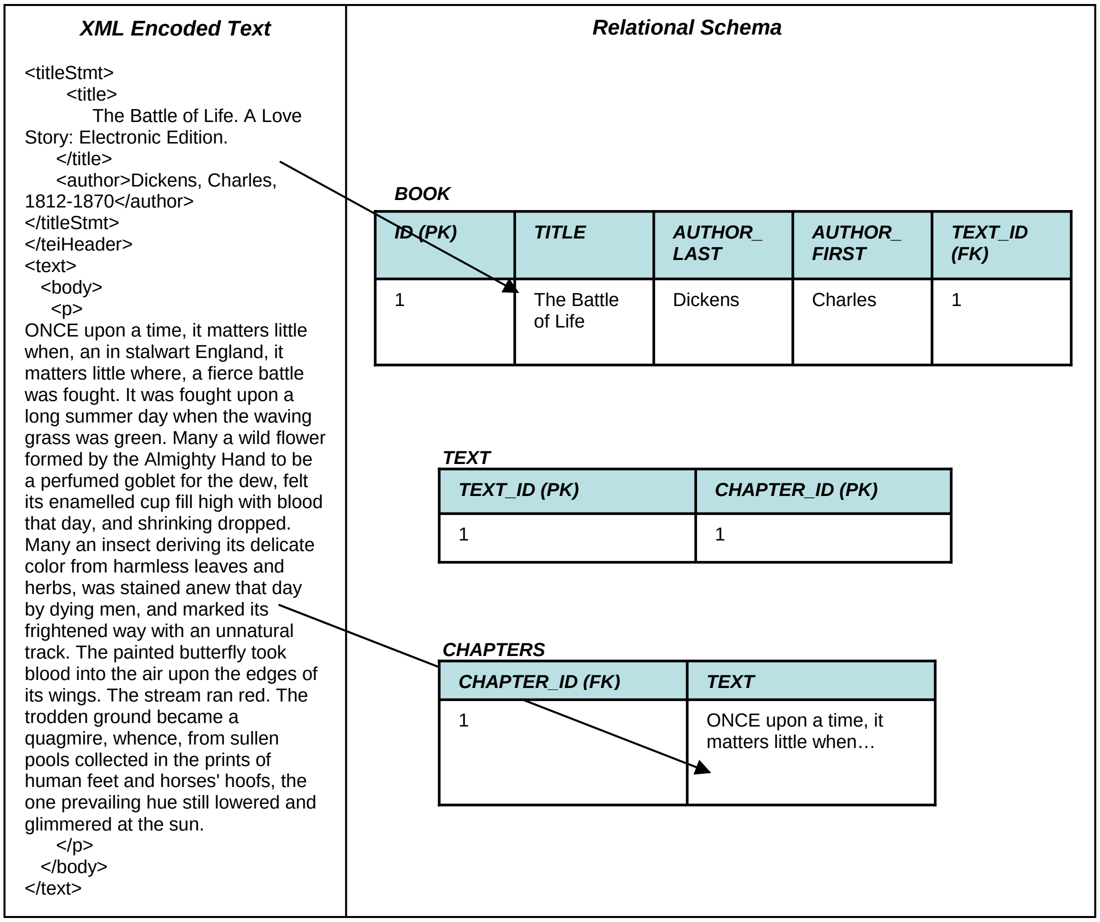
An interesting question brought up in the Kotsakis (2002) study and alluded to in historical research on text retrieval, concerns retrieval units for structured documents. What constitutes a document in a XML collection? Is a document considered the element or attribute in the XML file, or is the file itself the unit of retrieval? Moreover, should a XML retrieval model offer the end-user the opportunity to search each XML tag as a separate document? And, if so, how would the user have prior knowledge of this structure? Should end-users be able to dictate the appropriate unit through query expansion and relevance feedback?

Passage retrieval, defined by Salton, Allan and Buckley (1993) provides a foundation for exploring these XML retrieval issues. The impetus for their investigation revolved around the idea that full-text is best searched and indexed by its logical parts, since “most text items are naturally sub dividable into recognizable units, such as text sections, paragraphs, and sentences” (p. 49). Analyzing documents and excerpting passages from full-text documents, Salton et al. found that “by using passage retrieval strategies designed to retrieve text excerpts of varying sizes in response to statements of user interest,” users may find more precise and relevant information (p. 49).

The ideas from Salton et al. (1993) about passage retrieval apply directly to this discussion and could prove valuable techniques to developers and researchers building XML retrieval languages and systems. Large electronic encoded XML book collections are characterized by structural units such as table of contents, chapters and full-text sections. These elements could easily be excerpted and presented to the end user during the search process, thereby alleviating what Salton et al. refer to as “user overload” (p. 52). This phenomenon can be avoided “and the retrieval effectiveness enhanced by making it possible to retrieve text passages instead of full documents only…" (p. 53). Passage retrieval could guide the user through large documents; they could specify which paragraphs, table of contents or themes they deem both relevant and important.

A recent article by Carmel, Maarek, and Soffer (2001) looks at various methods of XML retrieval units that echo the techniques researched by Salton et al. (1993) and with previous research on semi-structured, document-centric XML retrieval (p. 151). Carmel et al. write that non-relational database techniques are emerging and that there is a "departure from the classical pure 'database' view of XML" (p. 151). Looking at "XML fragments" in relation to a vector-space IR model, the Carmel et al. study defines queries based on structural and content-based fragments of text (p. 153). Furthermore, their model extends Kotsakis' (2002) inverted indexing scheme to "use as indexing units not single terms but pairs in the form (*t*,*c*), where a term *t* is qualified by the context *c* in which it appears" (p. 153). The results of the Carmel et al. study resulted in high precision, leading the researchers to conclude that querying XML documents using an IR based approach results in highly precise search results (p. 155).

Current XML retrieval research contributed by the IR community provides a solid foundation for the emergence of systems that use document structure as a factor in the search process. Many researchers have shown that the knowledge and use of XML or SGML elements in searching and indexing greatly affects the precision of the IR system. Future work is necessary, however, to determine the best methods for building and evaluating such systems. More research is needed to address how such hybrid systems compare with classical keyword based techniques. The research presented here is a start in that direction.



XML Encoded Text

```
<titleStmt>
       <title>
            The Battle of Life. A Love
Story: Electronic Edition.
     </title>
     <author>Dickens, Charles,
1812-1870</author>
</titleStmt>
</teiHeader>
<text>
  <body>
    <p>
ONCE upon a time, it matters little
when, an in stalwart England, it
matters little where, a fierce battle
was fought. It was fought upon a
long summer day when the waving
grass was green. Many a wild flower
formed by the Almighty Hand to be
a perfumed goblet for the dew, felt
its enamelled cup fill high with blood
that day, and shrinking dropped.
Many an insect deriving its delicate
color from harmless leaves and
herbs, was stained anew that day
by dying men, and marked its
frightened way with an unnatural
track. The painted butterfly took
blood into the air upon the edges of
its wings. The stream ran red. The
trodden ground became a
quagmire, whence, from sullen
pools collected in the prints of
human feet and horses' hoofs, the
one prevailing hue still lowered and
glimmered at the sun.
     </p>
  </body>
</text>
```

Relational Schema

BOOK

ID (PK)	*TITLE*	*AUTHOR_ LAST*	*AUTHOR_ FIRST*	*TEXT_ID (FK)*
1	The Battle of Life	Dickens	Charles	1

TEXT

TEXT_ID (PK)	*CHAPTER_ID (PK)*
1	1

CHAPTERS

CHAPTER_ID (FK)	*TEXT*
1	ONCE upon a time, it matters little when…

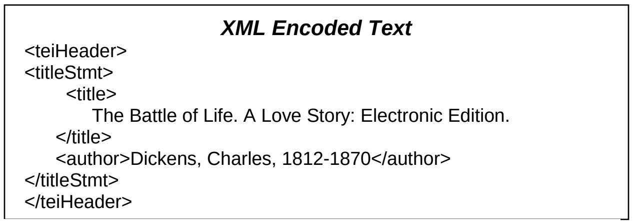
Figure 1.0
Simplistic mapping from a document-centric XML file to a relational schema

2.3.2 Relational & Native Databases & XML Retrieval

Like the IR community, there has been a recent surge of research and development efforts by database vendors to explore XML retrieval from a document-centric perspective. Until recently, vendor efforts have been primarily focused on data-centric, relational methods of retrieving and storing documents. These efforts have generally included storing XML via complex mappings between DTDs and relational tables. Figure 1.0 depicts a simplistic mapping of XML data to an Oracle database from the data-centric perspective using an excerpt of a XML file from Charles Dickens' collection, a digital library sponsored by Documenting the American South (DocSouth). XML elements and attributes are mapped to corresponding tables and fields. Text could be broken into chapters as shown in the diagram or copied into a single field. It is quite clear in either approach, however, that even simplistic mappings could quickly become overly complex and unmanageable.

Another disadvantage to using XML to relational database mappings is that once the data is mapped to relation structures, the integrity of the “document” is lost. To address this aspect of XML storage and retrieval, database vendors such as Oracle have introduced more document-centric solutions for storing and querying large full-text XML documents (Gennick, 2003). The Oracle XML DB Repository solution is an attempt by database vendors to “bridge the gap between the hierarchical, document-centric world of XML and the tabular, set-oriented world of relational databases” (para. 1). The Repository allows end users to store XML using both relational and native XML methods. Rather than mapping entities to corresponding fields in a relational database, the entire text of the file is stored in a Character Large Object datatype (CLOB). The document is kept intact as its XML tags are not separated into relational fields. Once the “XML documents are in the repository,” they are available “in either an XML-centric or a relational-centric manner” (para. 2).

Although Oracle’s Repository with CLOB storage seem like viable solutions to XML retrieval challenges, there are several disadvantages to using this type of technology.
A CLOB object does not provide the fine-grained detail that results from inverted files and other keyword based indexing techniques used in document databases. Methods of accessing these objects can be quite cumbersome. Oracle has added XPath extensions to their Structured Query Language (SQL) (see Figure 2.0) in an effort to provide



XML Encoded Text

```
<teiHeader>
<titleStmt>
      <title>
          The Battle of Life. A Love Story: Electronic Edition.
     </title>
     <author>Dickens, Charles, 1812-1870</author>
</titleStmt>
</teiHeader>
```

Figure 2.0
Retrieving values from an Oracle CLOB object using SQL and XPath in Oracle's SQL*Plus client (from Dillon, 2003)

relational/document-centric access to the object; however, there is limited “accessibility to SQL features” and the queries can quickly become overly complex (Dillon, 2003, Table 1). Furthermore, CLOB objects often produce “mediocre performance for data manipulation language (DML)” and “can consume considerable space,” making these solutions less attractive if server space is an issue (Dillon, 2003, Table 1).

At the same time as database vendors like Oracle are complementing their relational systems with document-centric features, the open source software community is actively building systems and languages to access document-centric XML collections. Native XML databases (NXDs), systems that store XML in a full-document format, incorporate many of the indexing techniques and querying mechanisms found in document database systems. Systems such as Apache Xindice (2003), Exist (2003), Tamino by Software AG (2003), and Berkeley DB by Sleepycat Software (2003) have included inverted indexes in their architectures that use the path of the XML document (elements and attributes) to access relevant information. These indexing schemes give developers and end-users more control over searching by allowing them to specify which paths of the document to search. Because the NXD system is easy to implement and maintain, there are many who believe that these systems will become more attractive to document-centric XML end-users and developers than the data-centric, relational methods proposed by large database vendors.

As with IR researchers, the open source and database vendor communities are actively pursing XML retrieval solutions. It is predicted that developing robust systems to take advantage of XML's ability to describe data will remain at the forefront of the research community for some time to come. NXD systems symbolize the move toward a fully integrated XML retrieval approach. This technology may provide end-users and developers with techniques to exploit XML tags in an effort to increase retrieval effectiveness in large collections of document-centric documents.

2.3.3 XML Retrieval & Online Search Engines

Despite recent discoveries in XML retrieval research, the search engine community has been relatively slow to enhance their algorithms to account for document structure. Search engines today typically employ keyword based searching that may produce high recall, but in many cases, low precision. These search engine systems provide little in the way of context or semantic based searching. As Berners-Lee, Hendler and Lassila (2001) point out, internet users need "a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported onto the web" in order for a true semantic and knowledge based web to be realized (Knowledge Representation Section, para. 14). They remark that "adding logic to the Web" through the use of Resource Description Framework (RDF) and XML provides "rules to make inferences" about information (Knowledge Representation Section, para. 15). Furthermore, Berners-Lee et al. (2001) stress that using a combination of RDF and XML will enable more sophisticated, content-based searches on the internet. These languages "ensure that concepts are not just works in a document but are tied to a unique definition that everyone can find on the Web" (Knowledge Representation, para. 18).

The vision of a semantic web outlined by Berners-Lee et al. (2001) is not fully embraced by search engine creators, however. Search Engine Watch, a group that monitors current news in the search engine world, writes that there has been a underlying tension between the creators of semantic markup languages such as XML and RDF and search engine developers who feel that they "have not participated in the development of a new framework" (Sullivan, 1997, para. 19). Sullivan (1997) writes that search engines ignore metadata in web documents because they do not trust the authenticity of it (para. 19). The search engine community has experienced "that people lie, mislead, or do whatever they can to get on top" of the search engine result lists (para. 27). These facts underscore the difficulty of searching metadata rich documents on the internet and illustrate why the search engine community continues to provide end-users with systems based primarily on traditional IR techniques such as keyword searching.

Although further collaboration between large search engine creators and metadata authors is needed before online XML retrieval systems can be fully realized, there are developers and researchers investigating how a "structure-aware" search engine might look. A novel search engine prototype, XYZfind from Egnor and Lorde (2004), indexes and queries documents on their structure and content in order to provide internet users with enhanced semi-structured information retrieval on the web. The system "uses reactive and automatic techniques to add structure to initially unstructured queries" (Adding Structure to Search, para. 6). Developers have extended the traditional IR inverted index "to first associate keywords and XML path specifications where those keywords appear; (keyword, path) tuples are then associated with documents and locations…" (Extending the Inverted Index Section, para. 12). Egnor and Lorde (2004) note that the XYZfind system takes into account the fact that “XML has the ability to represent the semantics of data in a structured, documented, machine-readable form,” a concept that is the foundation of the semantic web (Introduction, para. 1).

By utilizing XML tags in indexing and query formulation, Jackson and Gilstrap (1999) believe that search engines can provide a better “roadmap to information” (p. 316). These retrieval systems could marry content-based retrieval technologies based on IR methodologies and relational database techniques to create more “semantically-aware” information repositories. They (1999) predict that the emergence of more sophisticated, robust XML retrieval systems could help to “transform the Web from a universal information space into a knowledge network” (p. 320).

3.0 Significance of the Current Research

Increasing interest in XML retrieval from IR, database and search engine communities is understandable given the widespread use of XML on the internet and in the business sector. A recent 2003 study by Mignet, Barbosa and Veltri that surveyed the growth of XML usage, found that “the ‘.com’ and ‘.net’ domains combined contained 53% of the documents and 76% of the volume of XML content on the Web” (p. 501). Researchers in all sectors are recognizing XML as a viable language to express both document structure and content. XML is no longer seen as only a tool for modeling transitory information. Rather, it is already becoming the predominant language for the publishing community and many researchers, such as Jackson and Gilstrap (2002), believe that it will emerge as the *lingua franca* of the internet.

Myriad for-profit and non-profit organizations could benefit from XML retrieval technologies. Miller (2000) states that several Digital Libraries (DLs) have already adopted XML based DTDs such as Text Encoding Initiative (TEI) or Encoding Archive Description (EAD) for collection description (e.g. DocSouth, 2003; Southern Folklife Collection, 2003). While many of these DLs have useful hierarchical finding aids (e.g. DocSouth's subject index: http://docsouth.unc.edu/subject_table.html or UNC's Southern Folklife Collection: http://www.lib.unc.edu/mss/sfc1/sinv.html), their structures help only in the browsing process and do not address the problem of searching document collections in a precise, exhaustive manner.

Traditional libraries will also look for viable solutions to effective XML retrieval in the coming years. As Integrated Library System (ILS) vendors begin to incorporate the language into their OPACs, it will be necessary for libraries to adopt XML content management systems and repositories. Miller (2000), a Systems Librarian at Stanford University, writes that, "a fully XML-based integrated library system is feasible within three to five years" (p. 22). ILS vendors are already researching and developing XML-based OPAC solutions (p. 22). The ILS vendor community (as well as libraries) will most likely look to IR research for answers about how to best search XML. Clearly, keyword searching may not meet all of their patrons' needs.

The need for research into XML retrieval technology is also evident in the business community, where companies are already confronted with XML retrieval challenges. Popular trade journals are predicting "potential nightmares" when it comes to retrieving document-centric XML via RDMBS methods such as SQL. While businesses have been primarily concerned using XML as a data-exchange, they are recognizing the need to use XML in more document-centric ways, such as describing internal manuals (Liotta & Preimesberger, 2003). What this means is that the current data-centric XML methodology of mapping XML tags to relational tables will become too cumbersome and complex for users (Liotta & Preimesberger, 2003). Businesses will need retrieval methods suitable to searching both the structure and content of their internal collections.

Current literature as well as research and the development of XML IR systems, point to a growing interest in many technology communities for alternative query languages and systems that can effectively search through XML document collections. Questions in all communities center on whether classical inverted file techniques and keyword querying will be sufficient for searching through XML documents. How do these methods compare with novel technologies such as NXD's? Could knowledge of document structure in query formulation actually improve search results? By comparing and contrasting structure and content-based querying and indexing methods with traditional keyword search engine technology, the current study explores these important questions affecting for-profit and non-profit stakeholders now and in the future.

4.0 Methods

4.1 Research Design

The current study was conducted to test the hypothesis that searching both structure and content in XML documents works to increase precision and recall over keyword based methods used in traditional IR systems. The methodologies utilized in this study were derived from a combination of TREC (2000) and INEX (2002) guidelines as well as Tague-Sutcliffe's (1992) paper on information retrieval experimentation. Table 1.0 outlines the experiment design:

2 Different IR Systems

Queries translated from subject specialist's recommendations

		Exist XML Database		Lucene Search Engine	
		precision	recall	precision	recall
Queries (1-15) against 95 docs	Simple Form of Query	XPath Queries over 95 documents		Keyword Queries over 95 documents	
	Complex Form of Query				

Table 1.0
Experimental Research Design

The experiment is based on a within-subjects design, with the subjects being 15 queries submitted to 2 separate IR systems, across a document collection of 95 XML/TEI encoded texts. Two systems were incorporated into the study as few products have built-in options for both content and structure indexing and querying. Thus, the researcher chose systems that could perform content and structure-based searching and indexing.

The first system chosen, Exist, a Java based NXD, indexes and searches through XML documents in a manner that considers both the document structure and content. The system is typically well suited to "applications dealing with small to large collections of XML documents which are occasionally updated" (Meier, 2002, p. 169). Exist differs from other XML based systems in that it stores XML documents in their native format rather than mapping XML tags to relational schemas. Documents are logically stored, indexed and searchable through user-created, hierarchical tree-based collections (p. 169). XPath query language is the primary mode of searching through documents in the Exist system. Other system characteristics that make Exist a suitable choice for the present experiment are: its accessibility (it is an open source product, easy to install and to index XML documents); its XPath implementation; and its novel indexing scheme, that is fine-tuned to handle large document-centric XML encoded texts.

The second system chosen by the experimenter, Lucene, provides contrast to the first as it uses traditional IR indexing algorithms that ignore most document structure. As a classical IR search system, Lucene “offers two main services: text indexing and text searching” (Gospodnetic, 2003, p. 1). Similar to other IR systems, it provides indexing and searching based on the content of the document through inverted file structures and keyword based searching. The system also extends the classical IR inverted file model in order to handle some structured indexing and querying. Lucene allows user-defined fields to be created during the indexing and searching process. These “fielded searches” allow the system to utilize document structure in a way similar to Exist and other NXD systems. The fact that both Lucene and Exist handle full-text as well as structure-based searching and indexing, makes these systems attractive and applicable to the current experiment. Major features available in both of these systems as well as how these options are utilized in the present study, are summarized in Table 2.0.

System Characteristic	Characteristic Description	Exist XML Database	Lucene Search Engine	Used in the Current Study
Query Phrases:	Query phrases are typically constructed with consecutive words surrounded by quotes, such as “information retrieval”	Supported using either the XPath contains() operator or the near() operator (e.g., //text()[near(., “information retrieval”, 0)])	Supported using quotes like “information retrieval”	Yes
Boolean Operators:	“Boolean operators allow terms to be combined through logic operators” (Jakarta Lucene FAQ, 2003)	Exist extensions: &= , |= operators, plus XPath Boolean operations (and, or, not)	AND, OR, NOT, +, -	Yes
Grouping with Boolean Operators:	The ability to group certain words or phrases together in with parenthesis, such as (information AND retrieval) OR discovery	Not supported	Supported	No
Collection and Document Operators:	Only applicable to Exist. Defines where to search – over documents or collections	collection() and document()	N/A	Only collection() to specify the Dickens collection; all documents are queried
Stemming:	“A program or algorithm which determines the morphological root of a given inflected (or, sometimes, derived) word form -- generally a written word form” (FOLDOC, 1993)	Supported	Supported with the Porter Stemmer Analyzer	No
Stop Word Removal:	The removal of common words such as: *a, an, the* from indexing and searching	Supported	Supported	Yes The SMART stop word list from TREC was implemented into the current study.
Fielded Searches:	In indexing, a name/value pair representing a keyword and a part of the document. In searching, the ability to specifically define fields or indexes on which to search.	Supported using the paths to the indexed part of the document (XPath query language)	Supported using searches such as: author: “Charles Dickens”	Yes

System Characteristic	Characteristic Description	Exist XML Database	Lucene Search Engine	Used in the Current Study
Indexing Schemes:	How the database or file set is indexed internally for speedy searching.	Numbering Scheme	Inverted Files	Yes
Wildcard, fuzzy, proximity, regular expression or range searches:	Series of extensions that allows users to search for inexact terms such as: data*, which will find data mining, database, etc.	Supported	Supported	Only range searches for dates in the Lucene queries
Boosting Query Terms:	The ability to assign a “boost factor (a number) at the end of a term” to increase the relevance of that term. “Higher boost factors indicates higher importance of the matched construct” (Jakarta Lucene, FAQ, 2003).	Not supported	Supported	No
Query expansion or synonyms:	The “process of…adding search terms to a user’s weighted search” using thesauri or other synonym lists. (FOLDOC, 1993). The goal is to achieve greater recall and precision in querying.	Not supported	Supported	No

Table 2.0
Exist & Lucene System Characteristics

4.2 Testing Methodologies

To conduct the current experiment, the researcher implemented a 2 step approach. Test I served as the baseline for Test II. No indexes were altered or updated for this baseline phase. In Exist, every tag in the document collection was indexed according to an Exist internal numbering scheme. In Lucene, all text in the document collection was indexed using the full-text indexer. Tagging structures were ignored as tags; rather they

XML Field Name	Field(s) Mapped to in Lucene	Path(s) Mapped to in Exist	Description
<author>	author	//biblFull/titleStmt/author	The author of the book
<title>	title	//biblFull/titleStmt	The title of the work
<publisher>	publisher	//biblFull/publicationStmt/publisher	Who published the work
<date>	pubDate	//biblFull/publicationStmt/date	The date that the original edition was published
<pubPlace>	pubPlace	//biblFull/publicationStmt/pubPlace	Location where the original edition was published
<name>	illustrator	//div1[@type &="illustration*"]/p	Name(s) of illustrators
<note>	callNo	//biblFull/titleStmt/notesStmt/note	The Rare Books Call No
<edition>	edition	//biblFull/editionStmt/edition AND //docImprint	Edition of the original work
<text>	bookText	//text	The fulltext of the work

Table 3.0
Mappings from XML to Lucene and Exist used in Indexing

were indexed as keywords. Inverted files were created by the software to represent keyword-value pairs in each XML document.

Query results from Test I were compiled and evaluated using manually generated relevance judgments and classical precision and recall IR performance measures. The Test I phase of the experiment resulted in baseline precision and recall values that were compared with Test II results.

In Test II, the researcher manipulated each system in an effort to exploit XML tagging structures. To begin this phase of the experiment, 9 bibliographic indexing fields were chosen. Five of the fields were modeled after common bibliographic indexes found in many library information systems. Other fields were added after examining the most common fields used in the test queries. Additional fields included: call number, illustrator and edition. Table 3.0 lists the entire set of fields and how they were mapped to the document structure in each system.

Once fields were selected, each system was altered to include the chosen indexes. To update indexes in the Exist system, it was necessary to edit the index in conf.xml. Figure

```
<!—Stemming is turned off; Indexing is case insensitive;
indexing only at a depth of 3 nodes using the simple tokenizer
-->

<indexer stemming="false" caseSensitive="false" suppress-
whitespace="both" index-depth="3"
         tokenizer="org.exist.storage.analysis.SimpleTokenizer"
         validation="false">

   <!—File containing a list of stopwords to be ignored by the
parser. -->
   <stopwords file=" /exist/english.stop "/>

  <!—Indexing configuration for Test 2 -->
      <index attributes="true" doctype="TEI.2" default="none">
             <include path="//biblFull/titleStmt/author"/>
             <include path="//biblFull/titleStmt"/>
             <include
path="//biblFull/publicationStmt/publisher"/>
             <include path="//biblFull/publicationStmt/date"/>
             <include
path="//biblFull/publicationStmt/pubPlace"/>
             <include path="//div1"/>
             <include path="//biblFull/titleStmt/notesStmt/note"/>
             <include path="//biblFull/editionStmt/edition"/>
             <include path="//docImprint"/>
             <include path="//text"/>
      <include path="//TEI.2"/>
      </index>
</indexer>
```

Figure 3.0
Excerpt from the Exist "conf.xml" indexer configuration file

3.0 depicts an excerpt from the indexer portion of the configuration file. As shown in the excerpt, stemming is turned off, indexing is case insensitive and a list of English stop words is used in indexing the documents. A list of paths to include in indexing, corresponding to the 9 bibliographic files in Table 3.0, was added to the configuration file. The root node is also included in the path list to retain full-text searching capabilities and to remain consistent with the Lucene system. Thus, either fielded searches or full-text searches could be submitted to the Exist indexer in Test II. Once the configuration file was updated for the Exist database, the test document collection was reindexed to reflect the new changes.

To alter the Lucene system for Test II, *Field* objects corresponding to bibliographic fields in Table 3.0 were added to the *IndexFiles* Java class. The document collection was then reindexed to include bibliographic fields as well as the full-text of the document for searches that do not specifically use bibliographic fields.

4.3 Document Collection

The testbed for the current experiment consisted of a collection of 95 XML/TEI encoded documents. These documents were made available through the DocSouth (2004) project, a digital library initiative at the University of North Carolina at Chapel Hill. The works and their titles are listed in Appendix A.

File Summary	
Total Size of Collection	21.55 MB
Avg. File Size	0.23 MB
Minimum File Size	0.01 MB
Maximum File Size	1.89 MB
Avg. Number of Elements	2050
Avg. Number of Attributes	3939
Most Used Attribute	<p>
Maximum Elements/Document	9845
Maximum Attributes/Document	18808
Minimum Elements/Document	162
Minimum Attributes/Document	222

Table 4.0
Summary of the Document Collection

Each document in the collection is both XML and TEI compliant. To ensure XML compliance, the documents were validated by an XML parser, Xerces. The parser checked for well-formed XML, meaning that a XML document is syntactically correct. The syntax rules for XML are defined by the W3C Schools (1999) and include guidelines such as ensuring that each tag has a corresponding closing tag and checking that attributes are properly quoted.

To ensure TEI compliance, each document must conform to specifications of a Document Type Definition (DTD) as specified by “XML Validation” document from the the W3C Schools (1999). The documents included in this study are validated against the *teixlite* DTD which can be found at: http://www.tei-c.org/Lite/DTD/teixlite.dec. The system that was used to perform XPath searches in this study parsed each document to ensure both well-formed and valid XML.

The document corpus used in the study consists of fairly in-depth tagging structures that correspond to the main elements outlined by the Text Encoding Initiative (TEI) group (1999). The most heavily used element in the test collection is the paragraph tag (<p>). The average number of nodes/elements in the documents is 2050 and the average number of attributes is 3939. Table 4.0 summarizes element and attribute sizes for the document collection.

4.4 Query Formulation

To conduct both tests, queries relevant to the Dickens document collection were submitted to each system. Each query was formulated initially with the help of a subject matter expert in the Rare Books Department of the Wilson Library at the University of North Carolina at Chapel Hill. Informal meetings with the Rare Books Librarian resulted in 15 natural language queries, representative of common inquiries from library patrons. Each natural language query was subsequently translated into a simple and complex form, using both XPath query language and keyword phrases. For example, the natural language question: “Does the library own any ‘Diamond Editions’ in the Charles Dickens collection?” was translated into 4 separate queries:

- Simple XPath Query: //TEI.2[near(., 'diamond edition')]
- Complex XPath Query: //biblFull/titleStmt/title[near(., 'diamond edition')]
- Simple Keyword Query: "diamond edition"
- Complex Keyword Query: contents: "diamond edition"

The simple XPath form of the queries performs full-text search from the root node of the XML document. Likewise, the simple keyword query searches through the document space only on its content rather than using any document structure. Until the system was altered to include fields in the Lucene system, the keyword queries disregard the structure of the document. Complex queries for both systems consider document structure, however. The XPath version uses path-based nodes to point directly to the content of the document, whereas the complex keyword phrase uses field syntax to point at fielded indexes. Appendix B provides a complete list of the natural language queries and their respective translations for both systems tested in the study.

Exhaustive relevance judgments for each query were created by the researcher and reviewed for accuracy by the Rare Books Librarian at Wilson Library. This expert was also the same person that provided insight in the creation of the natural language queries for the current project. Relevance judgments were used in calculating precision and recall numbers for each query, for each system, at each test. Appendix C lists the relevance judgments for each query used in the present study.

4.5 Performance Measures

Query results from the 2 tests were calculated using IR precision and recall performance measures. Defined by Salton and McGill (1983) in *Introduction to Modern Information Retrieval*, precision is “the ability [of a system] to present only the relevant items” and recall is “the ability of the system to present all relevant items” (p. 162). In the current experiment, precision and recall were computed for each set of query results and calculated using a *recall-precision matrix* outlined in Chowdhury (1999).

4.6 Testing Criteria & Assumptions

The design of Test I and Test II was based on the following specifications and assumptions.

- In Test I, fields added to the complex queries in Lucene were ignored; bibliographic indexes were not specified explicitly in the Lucene system until Test II.
- Stop words were removed in both the indexing and querying process of Exist and Lucene. The SMART stop word list was chosen as the default in both experiments (SMART Tutorial).
- The StandardAnalyzer class was incorporated into both tests in Lucene. The class constructor allows a user-defined list of stopwords to be specified.
- All dates encoded in Roman Numerals were manually translated to numeric dates for the purposes of searching and indexing, as neither system had a way to translate these dates.
- Both tests were conducted on the same Linux RedHat 9 operating system. The Lucene and Exist product software was located on the same machine as well. A complete listing of the hardware, software and indexing configuration is in Appendix D.

- Range operators: *[x TO y]*, available in Lucene were only used in the complex date field of Query 7 for Test II, as it was deemed appropriate to use the technology to grab relevant data.
- The XPath query for Query 7 also utilized XPath specific range features such as the >= operators.
- The default field in Lucene for full-text indexing and searching is *contents:* in both Tests and in all query runs. Lucene does not require this field to be specified explicitly in queries.
- Full-text searching support in Exist is implemented through the XPath *contains()* function and through the Exist specific *near()* function. The *near()* function was used in many simple queries for Test I in the Exist system, as it is the only method in which to search using keyword phrases. Additionally, the Exist (2003) documentation states that it is exponentially faster than the *contains()* operator and provides the same functionality.
- Because Lucene does not recognize XML tagging structures, it is necessary to feed "chunks" of XML to the indexer for field structures used in Test II. Due to the fact that the XML files in this collection have numerous references to the tag <name> for example, and that Lucene does not recognize paths, it is necessary to index all tags known as <name> or <date> or any of the other indexed fields for Test II.
- Query timing of the individual queries was not calculated as a performance measure; only precision and recall were taken into account.
- Retrieval units for the present experiment are defined as the entire document rather than individual tags or sections of the document.
- Query ranking was purposefully left out of the experiment because of the alterations needed on each system to incorporate ranking. Thus, precision and recall were calculated at each change in query rather than as documents were retrieved from each system.

4.7 Limitations of the Research Design

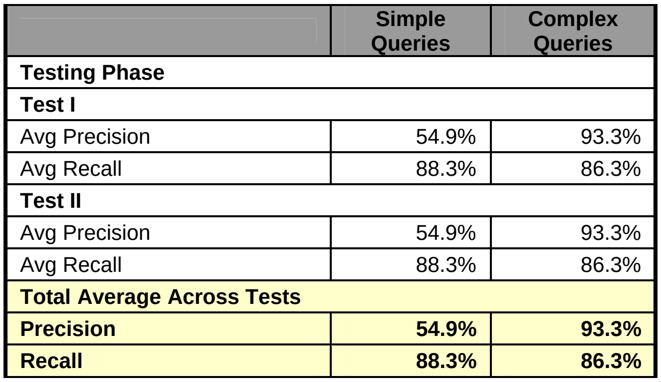
Library of Congress Subject Headings (LCSH) were not included in the original XML document collection, as the electronic editions of these works were never catalogued. To remedy this, the researcher searched through cataloguing records in WorldCat and the University of NC Chapel Hill OPAC for LCSH associated with the physical works corresponding to the electronic editions. The searches produced little LCSH of use to the present study. UNC Library cataloguers later confirmed that many works of fiction published prior to the 1970s were not catalogued in the universal systems. As a result of this finding, subject headings were excluded from the current experiment in both indexing and searching.

Another limitation to the current experiment was the exclusion of stemming and query expansion techniques typically used in IR systems. Because the experiment sought to look at system performance rather than query optimization, the researcher chose to exclude stemming features from both systems.

Other options available for query enhancements were also left out of the present study. For example, Lucene options for regular expression and "fuzzy" operators were not utilized. Proximity searching was not used in either system except in the case of keyword phrase queries in Exist, where the only way to search for phrases is through the *near()* or *contains()* operators. Wildcards were not incorporated into any queries in the experiment. Additionally, synonyms were not part of the study.

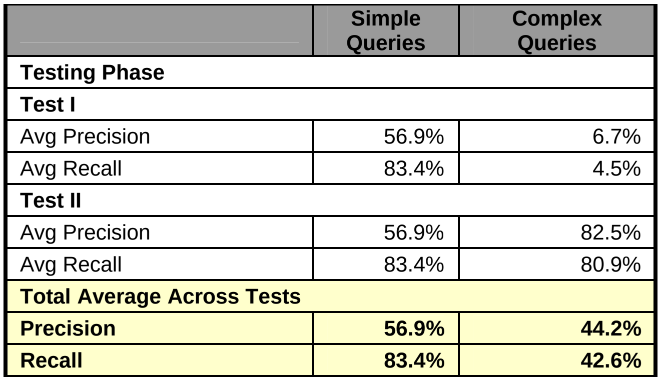
5.0 Results

Two systems were used in the present study to compare and contrast the effectiveness of adding XML document structure to simple and complex queries. Results were favorable in both phases of testing and revealed that on average, complexly constructed queries combining structural and content elements of a document produce greater precision than keyword queries. This is evidenced in Tables 5.0-6.0 and Figure 4.0, which provide aggregate views of the testing results. Figure 4.0 shows that simple queries for the Exist and Lucene systems produced relatively mediocre retrieval results while complex queries in all tests of the Exist product and in the second phase of testing of Lucene, produced remarkably high precision and recall numbers. Figures 5.0-6.0 show precision and recall at the query level for Test I and Test II across both systems for simple and complex forms of the queries. Appendix E details all precision and recall results at the query level in table form. Appendix F shows a series of descriptive statistics about each system for each test across both simple and complex queries.



	Simple Queries	Complex Queries
Testing Phase		
Test I		
Avg Precision	54.9%	93.3%
Avg Recall	88.3%	86.3%
Test II		
Avg Precision	54.9%	93.3%
Avg Recall	88.3%	86.3%
Total Average Across Tests		
Precision	**54.9%**	**93.3%**
Recall	**88.3%**	**86.3%**

Table 5.0
Exist Precision & Recall Results



	Simple Queries	Complex Queries
Testing Phase		
Test I		
Avg Precision	56.9%	6.7%
Avg Recall	83.4%	4.5%
Test II		
Avg Precision	56.9%	82.5%
Avg Recall	83.4%	80.9%
Total Average Across Tests		
Precision	**56.9%**	**44.2%**
Recall	**83.4%**	**42.6%**

Table 6.0
Lucene Precision & Recall Results

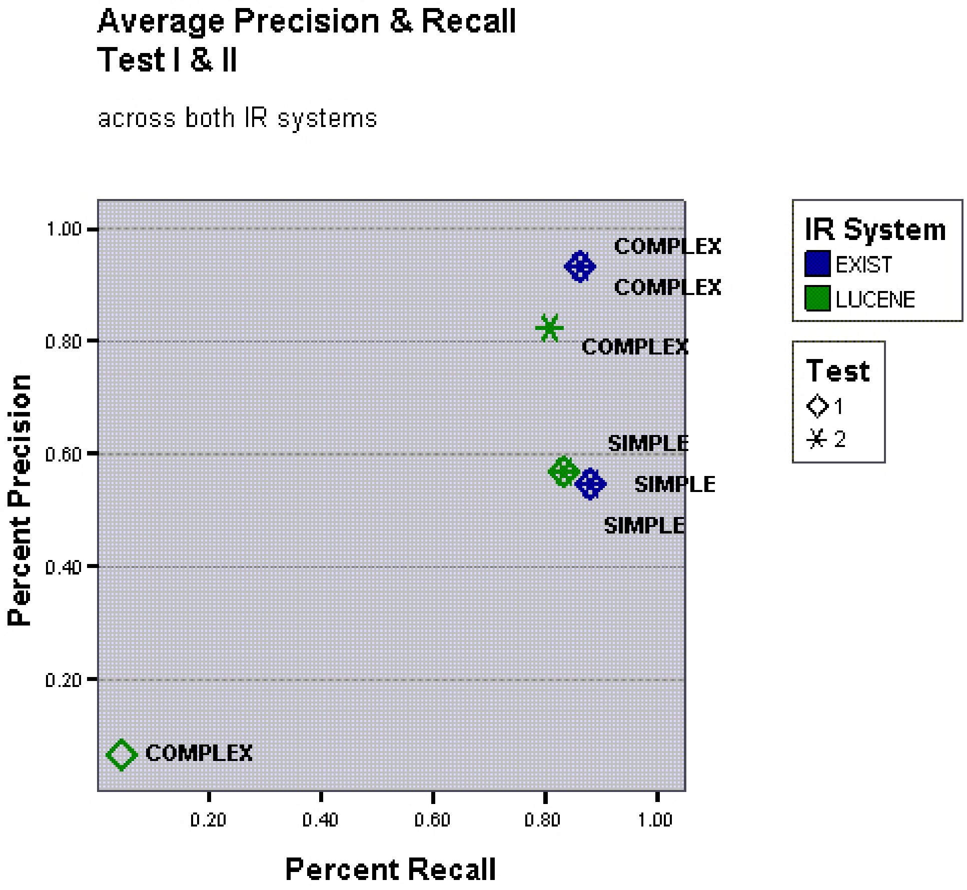


Figure 4.0
Average Precision & Recall for Test I & Test II

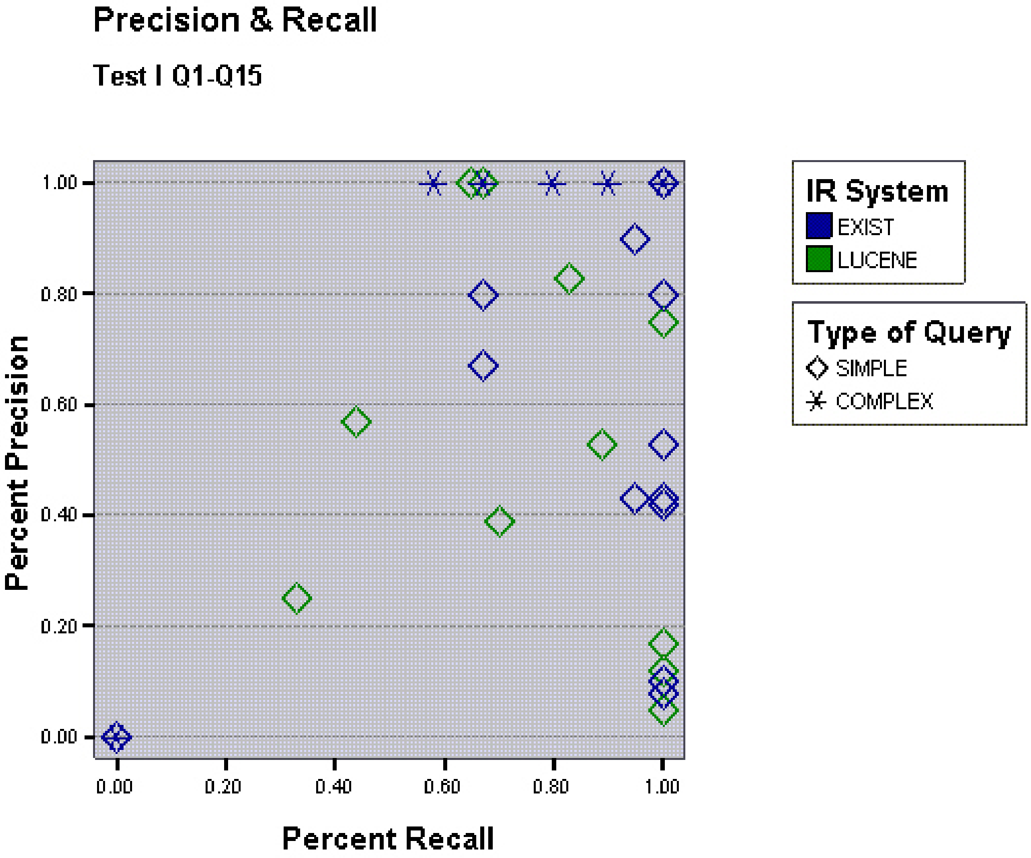


Figure 5.0
Precision & Recall for Test I for all Queries across both Systems

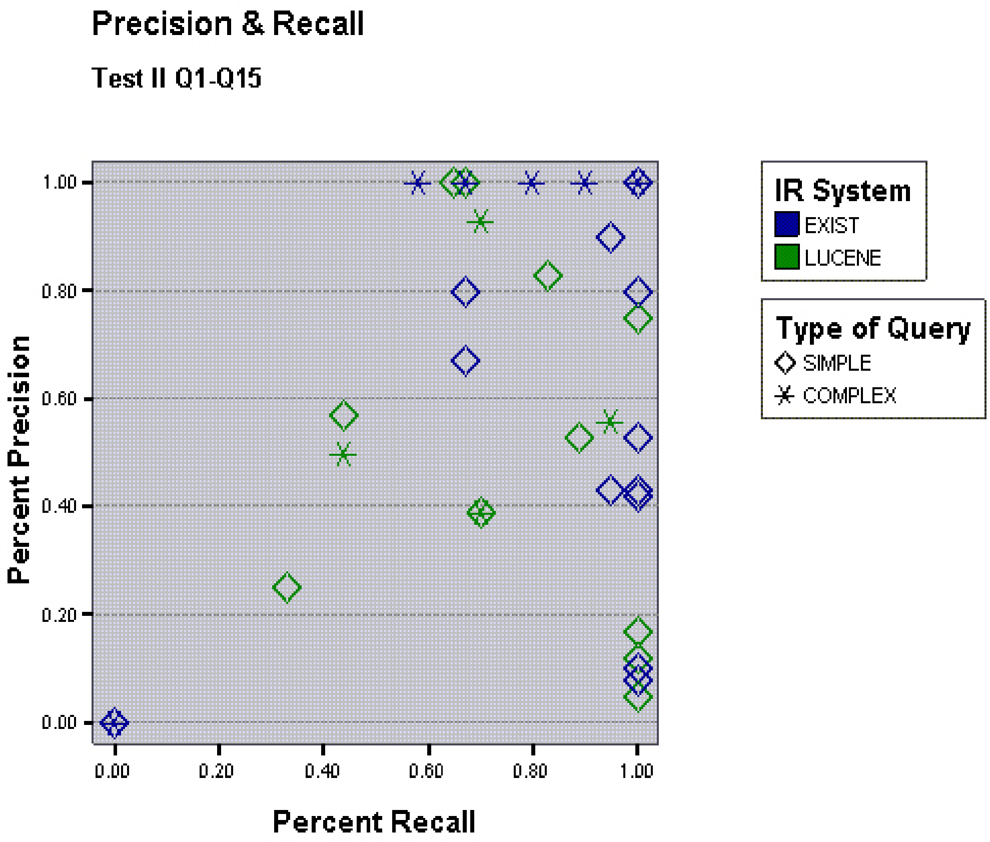


Figure 6.0
Precision & Recall for Test II for all Queries across both Systems

5.1 Phase I – Baseline Test Results

The first phase of testing for the present experiment served as a baseline for both the Exist and Lucene IR systems. As detailed in the Methodology section, a series of 15 pre-defined simple and complex queries were run against each system. The default indexing and server configuration for both systems was unaltered in this first phase of testing. Precision and recall numbers were calculated using pre-formulated relevance judgments.

Performance results from the Exist system in this first test indicate that there is a strong correlation between the complexity of the query and precision performance. This suggests that prior knowledge of document structure positively affects retrieval precision (See Figures 4.0-6.0). In Exist, both precision and recall were high for complex queries that incorporated prior knowledge of XML tagging structures. The system attained an average precision of 93.3% and an 86.3% average recall in the complex, more structured based queries. Complex queries produced only a slight decrease in recall performance between complex and simple queries as shown in Figures 4.0-6.0. This slight decrease could be explained in the inverse recall/precision relationship first detailed by Cleverdon (1972) and written about in most IR texts today. In both tests, this inverse relationship is visible; the more complex the query is, the higher the precision and lower the recall. An interesting aspect of the precision and recall relationship in this particular set of results is that there is only a slight difference between the averages of both performance measures (93.3% and 86.3%). In the aggregate, it is thus clear that recall is not compromised in the Exist system as results become more precise, an aspect of the system that could satisfy a wide range of users with various information needs, from those who want very precise answers to those who desire larger result sets.

At the query level, the Exist product produced unusually high performance percentages in this first phase of testing (see Appendix E). Queries 3-6, 10-12 and 15 resulted in 100% precision and recall, a phenomenon rarely witnessed in retrieval systems. The results might be explained by 2 different factors built into the experimental design:

1. the precise nature of the predefined XPath queries constructed, and
2. the precise nature of the manually generated relevance judgments.

In other words, because the researcher had prior knowledge of the document structure (the TEI/XML tags), she was able to formulate more precise queries, possibly leading to these high performance marks. Although these results might appear ideal and as a consequence, unrealistic, they do indicate how strongly prior document and collection knowledge weigh on retrieval results. Knowledge of a document's XML tagging structures may help end-users formulate more precise queries thus leading them to discover information more quickly and easily than provided in current keyword based searches.

Although Exist performed remarkably well on most test runs, there were several queries that produced unexpectedly low precision. Query 2, for example, returned no documents. These anomalous results can be explained by the fact that in query formulation, the researcher found inconsistencies in the encoding of the Dickens XML documents. The complex XPath form of Query 2 (//biblFull/titleStmt/title[near(., 'diamond edition')] only retrieved documents where "Diamond Edition" appeared under the "biblFull" node. The researcher found, however, that this was not the only place where "Diamond Editions" appeared. Some works had the "Diamond Edition" designation encoded in the head element of the XML document. Therefore, a query such as //head/hi[@rend="italics"][near(., 'diamond edition')] could have been combined with the “biblFull” XPath query above to retrieve all instances of “Diamond Edition.” With consistent encoding, documents would have been returned for Query 2.

The Lucene search engine performed as expected in the first baseline test. Simple queries resulted in lower precision values than in the Exist product (Appendix E). These results were attributed to the fact that keyword searches in many search engines produce high recall, but often low precision. In Test I, Lucene produced an average recall of 83.4%, but low precision at 55.3%. Interestingly, the recall and precision figures for Test I simple queries aligned with the results from Exist, leading the researcher to posit that the two systems are comparable in terms of their full-text indexing and keyword searching capabilities.

Complex query results for Lucene in the baseline test were skewed as outlined in the research design. Queries were purposefully kept consistent across Test I and Test II in an effort to reduce bias and to limit the scope of the study to measuring only the performance of the two systems. This design decision meant that in this phase of testing, Lucene was incapable of recognizing the “fields” specified in the complex queries because they were not coded in its index. Only when the indexes were altered in Test II, was Lucene able to recognize the searchable fields. Therefore, the weak complex query results for Lucene in Test I were expected and accounted for by the researcher.

5.2 Phase II – Indexing Manipulation Results

As described in the Methodology section, the independent variables for this experiment were the queries. The treatment imposed on each system in Phase II was the manipulation of each system's indexing through the addition of 9 bibliographic fields. In both systems, these indexing modifications had a positive effect on complex query results, but they were of little consequence to the simple query performance. In Lucene, for example, index manipulation did not affect the precision and recall of the simple keyword queries. It dramatically impacted the precision and recall of the complex queries, however. The system achieved an overall average precision of 82.5% for complexly constructed queries and an average recall of 80.9%. As expected, recall dropped slightly in the complex test runs from the simple runs where recall was 83.4%. This was attributed to the inverse relationship mentioned in the summary of Test I. Despite this slight shift, however, Lucene's performance points to the conclusions drawn in the discussion of Exist results, namely that recall was not considerably compromised by high precision. The average precision rate after fields were added to Lucene, suggest that structure can effectively help in the retrieval of content. More precisely, both the Exist and Lucene complex query results indicate that structure based queries and indices have a positive effect on precision performance.

Interestingly, while Lucene's performance improved with indexing enhancements, Exist was not affected at either the query or aggregate level by these changes. Both sets of Exist results remained consistent across both test runs. This could be explained by the way the Exist indexer works as compared to Lucene's. Manipulating the Exist indexer to include or exclude certain paths in the XML documents only filters these paths from the full-text indexer, not the built-in indexing scheme. As a result, the paths were still included in the indexing process making them accessible by all of the queries. Performance gains in the Exist system in Test II would most likely have been achieved by calculating the system performance based on query times, rather than using precision and recall. This statistic, however, was not part of the test design of the current experiment, although it could easily be incorporated into future research on this topic.

6.0 Discussion

In the current study, the researcher posited that a combination of document structure and content in query formulation and indexing enhances full-text precision and recall performance of online document-centric XML encoded texts. The favorable performance results attained from the Exist and Lucene systems support the theories put forth by researchers such as Baeza-Yates and Navarro (1996) and Luk et al. (2002), who argue that hybrid models, combining classical IR, relational database and full-text retrieval techniques, might result in optimal retrieval results for end users of information systems. The present research works to this end and has shown specifically that complex queries coupled with indexing enhancements produce greater precision and at the same time do not compromise recall performance. This research has also indicated that higher precision and recall is achieved by having prior knowledge of document structure and using this information in indexing and query formulation. By specifying detailed queries that correspond to paths in XML documents, the researcher has shown that an understanding of a document’s structure provides the user with a greater chance of retrieving relevant, precise information.

Although the research presented here is quite positive, it does elicit important questions regarding how information system designers might empower users with knowledge of the structure underlying a corpus of XML encoded documents. In other words, is it realistic to expect end-users, especially novice ones, to know and understand

XML tagging schemes? This question may best be answered by first understanding the user's search process. How the user forms queries and searches through a document collection is well-described in Haas' (2003) paper, *Improving the Search Environment: Informed Decision Making in the Search for Statistical Information*. She outlines two "dimensions" of a user's search environment:

- the dimension that "represents the knowledge that the user can bring to bear on the search"
- and the knowledge that the user has of the search process or "how to conduct a search" (p. 783).

Haas maintains that "the decisions that are made in a search are enabled and constrained by these two dimensions" (p. 796). She explains that an understanding of the dimensions may contribute to "the goal of augmenting the user's domain and search knowledge in the context of a particular configuration" (p. 796).

The two dimensions Haas (2003) describes relate directly to the results achieved in the current research. What was shown in both tests is that if users have an in-depth understanding of the domain or collection in which they are searching and if they are empowered with tools to effectively search through that domain, then it is quite probable they will find useful and relevant information that meets their needs. As Haas argues, an understanding of the user's search process is valuable information (p. 783). If designers are aware of and recognize the two dimensions or frameworks from which a user comes to an information system, they may be better equipped to provide the right tools to help the user formulate more complexly constructed queries. For example, a designer could account for the fact that a novice user approaches a document collection with no prior knowledge of it and little understanding about how to construct complex queries to search through it (Haas, 2003). Likewise, an understanding that expert users require advanced searching capabilities might help designers accommodate these needs as well. Techniques to address the needs of both novice and expert users are well-defined in the literature. For example, Alberg and Shneiderman’s (1994) work on data visualization may prove most useful for XML IR systems such as Exist by enabling the user to visualize the structure of a document before he/she forms their queries.

As evidenced by the results of the current experiment, to achieve high precision in XML IR systems, a user must be intimately familiar with the document collection, the structure of the document (in this case, TEI/XML) and the searching mechanisms of each system. An information systems designer should be familiar with the dimensions of the user’s search as explained by Haas (2003) to fully appreciate how best to model a document collection for the end user. Clearly, if the system is an online website, few users will approach it with an in-depth prior understanding of its contents. Only through a relationship between system designers and end-users can structure and content-based queries such as the ones used in the present study, be successfully combined to achieve better information retrieval in all types of information systems, including traditional IR, relational database and search engine repositories.

7.0 Limitations of the Current Study

As alluded to previously in this discussion, one limitation of the present study revolves around the fact that the researcher had an intimate knowledge of the document collection and was able to use this information to construct simple and complex forms of the natural language queries. This limitation caused the results to be skewed slightly in the Exist system as precision and recall figures often hit 100% at the same time at the query level. While a study limitation in some respects, these results strongly support the assertion that prior knowledge of document structure can positively affect retrieval performance when that information is used in indexing and querying. Moreover, domain knowledge about a collection works to increase query specificity. As outlined in the discussion, these limitations/factors of the study were not negative; rather, they illuminate what types of options must be considered when building systems that use both document structure and content.

8.0 Conclusion

The current study was conducted to demonstrate how exploiting document structure in indexing and querying improves retrieval performance of large document-centric XML texts. Despite the proliferation of IR, database and search engine literature on the topic of XML IR, little work had been done to compare classical keyword approaches with XML IR structured techniques. Likewise, few studies had been initiated to test the effectiveness of the myriad of XML retrieval products currently available. The present research was an effort to help bridge that gap. Results from the study revealed that using structure to augment queries over full-text XML collections enhances performance. Positive outcomes from two phases of testing provide concrete performance figures that are generalizable across a variety of domains. An effective XML IR system has the potential to provide users with precise search results in internet search engines, library OPACs, digital library collections, business content management systems or relational database text retrieval systems.

9.0 Future Research

This research has uncovered numerous areas for future study. First, more research is necessary to determine how to effectively design an IR system that can, at the same time, expose document structure and help end users in query formulation. Alongside this research, more study is needed to resolve how best to encapsulate the complex XPath query syntax. Clearly, end-users should not be required to construct such complex structures in order to retrieve relevant information.

Other possible avenues for future study include the incorporation of user interface design concerns brought up here into the design of an existing XML IR system, such as Exist. Key user studies could be conducted to help inform the design process and evaluate the performance of such systems.

Additional research should also look at indexing enhancements and query manipulation in the context of XML IR. For example, would Lucene have produced higher precision and recall if wildcard or proximity searches had been incorporated into this study? How would ranking have affected the precision and recall of each query?

Future research may also help determine whether coupling a full-text search engine like Lucene with a “structure aware” system like Exist might address XML retrieval challenges. Results from the present study suggest that search engines might still be quite suitable for simple keyword queries, while NXD’s may provide more fine grained access to highly structured document collections.

The topic of retrieval units is another area needing attention. Possible questions for future research might include: What retrieval units would be most appropriate for end users? If a term appears 3 times in a document, each instance occurring at a different

XML node, should the IR system return 3 hits or would the document count as 1 hit? Passage retrieval explained by Salton et al. (1983) could prove to be a valuable technique for this type of research.

Finally, different collections of both type and size could be evaluated to further explore structure and content-based XML IR. Using a larger collection from which a sample could be drawn may greatly affect the performance outcome of XML structured queries. Moreover, the types of collections may also influence precision and recall. For example, how would each system fare running queries over larger XML documents? Future studies that look at document collection types and sizes might also include user studies with input from both subject matter (domain) experts and system retrieval experts. Haas (2003) explains the differences between these two groups and their relationship with the systems in which they are searching. She cites Marchionini, Dwiggins, Katz and Lin (1993) as well as Fidel (1991a, 1991b, 1991c) as reputable sources for information on “the shared importance of domain and search expertise” (p. 785).

10.0 References

Ahlberg, C., & Shneiderman, B. (1994). Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. *Proceedings of the SIGCHI conference on Human Factors in Computing Systems: celebrating interdependence,* 313-317.

Apache Xindice. (2003). Retrieved November 2003 from http://xml.apache.org/xindice/.

Baeza-Yates R., & Navarro, G. (1996). Integrating Contents and Structure in Text Retrieval. *SIGMOD Record, 25(1),* 67-79.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern Information Retrieval*. New York: Addison Wesley.

Baeza-Yates, R., Fuhr, N., & Maarek, Y. (2002). Second Edition of the "XML and Information Retrieval" Workshop. *ACM SIGIR Forum, 36(2),* 53-57.

Baldi, P., Frasconi, P., & Smyth, P. (2003). *Modeling the Internet and the Web: Probabilistic Methods and Algorithms*. Hoboken, NJ: Wiley.

Barg, M., & Wong, R.K. (2001). Structural Proximity Searching for Large Collections of Semi-Structured Data. *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management (CIKM),* 175-182.

Berners-Lee, T., Hendler, J. & Lassila, O. (2001). The Semantic Web: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *ScientificAmerican.com.* Retrieved January 2004 from http://www.scientificamerican.com/print_version.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

Blake, G.E., Consens, M.P., Kilpelainen, P., Larson P.A., Snider, T., & Tompa, F.W. (1994). Text/Relational Database Management Systems: Harmonizing SQL and SGML. *Proceedings of the 1994 International Conference on Applications of Databases,* 267-280.

Bourret, R. (2003). XML and Databases. Retrieved October 2003 from http://www.rpbourret.com/xml/XMLAndDatabases.htm.

Bourret, R. (2003). XML Database Products. Retrieved October 2003 from http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

Carmel, D., Maarek, Y., & Soffer, A. (2001). XML and Information Retrieval: a SIGIR 2000 Workshop. *SIGMOD Record, 30(1),* 62-65.

Chaudhri, A.B., Rashid, A., & Zicari, R. (Eds.). (2003). *XML Data Management: Native XML and XML-Enabled Database Systems.* Boston: Addison Wesley.

Chowdhury, G.G. (1999). *Introduction to Modern Information Retrieval.* London: Library Association Publishing.

Clark, J., & DeRose, S. (Eds.). (1999). XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999. Retrieved November 2003 from http://www.w3.org/TR/1999/REC-xpath-19991116.

Cleverdon, C.W. (1972). On the Inverse Relationship of Recall and Precision. *Journal of Documentation, 23,* 195-201.

DeFazio, S., Daoud, A.M., Smith, L.A., Srinivasan, J., Croft W.B., & Callan, J.P. (1995). Integrating IR and RDBMS Using Cooperative Indexing. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 84-92.

DELOS. (2000). Network of Excellence on Digital Libraries. Retrieved November 2003 from http://delos-noe.iei.pi.cnr.it/.

Dillon, S. (2003). XML: To CLOB or Object? *Oracle Magazine*. Retrieved October 2003 from http://otn.oracle.com/oramag/oracle/03-jul/o43xml.html.

Documenting the American South (DocSouth). (2004). Retrieved November 2003 from http://docsouth.unc.edu.

Dodds, L. (2001). XML and Databases? Follow Your Nose. *O'Reilly XML.com*. Retrieved October 2003 from http://www.xml.com/lpt/a/2001/10/24/follow-yr-nose.html.

Egnor, D., & Lord, R. (2000). Structured Information Retrieval using XML. Retrieved January 2004 from http://www.haifa.il.ibm.com/sigir00-xml/final-papers/Egnor/.

Elad, B. (2000). XYZfind Goes Forward with XML Search Engine. *internetnews.com* Retrieved January 2004 from http://www.internetnews.com/bus-news/article.php/376311.

Exist Open Source Database. (2003). Retrieved August 2003 from http://exist-db.org/.

Exist Open Source Database. (2003). Server Configuration. Retrieved February 2004 from http://exist-db.org/configuration.

Extensible Markup Language (XML). (n.d.). Retrieved October 2003 from http://www.w3.org/XML/.

Fidel, R. (1991a). Searchers' selection of search keys: I. The selection routine. *Journal of the American Society for Information Science, 42(7),* 490-500.

Fidel, R. (1991b). Searchers' selection of search keys: II. Controlled vocabulary or free-text searching. *Journal of the American Society for Information Science, 42(7),* 510-514.

Fidel, R. (1991c). Searchers' selection of search keys: III. Searching styles. *Journal of the American Society for Information Science, 42(7),* 515-527.

FOLDOC. (1993). The Free On-Line Dictionary of Computing. Retrieved April 2004 from http://wombat.doc.ic.ac.uk/foldoc/index.html.

Fuhr, N., Govert, N., Kazai, G., Lalmas, M. (2002). INEX: Initiative for the Evaluation of XML Retrieval. *Proceedings of the SIGIR 2002 Workshop on XML and Information Retrieval,* 1-9.

Gennick, J. (2003). Make XML Native and Relative. *Oracle Magazine.* Retrieved October 15, 2003 from http://otn.oracle.com/oramag/oracle/03-jan/o13xml.html.

Großjohann, K., Fuhr, N., Effing, D., & Kriewel, S. (2002). Query Formulation and Result Visualization for XML Retrieval. *Proceedings ACM SIGIR 2002 Workshop on XML and Information Retrieval.* Retrieved February 2004 from http://www.is.informatik.uni-duisburg.de/bib/fulltext/ir/Grossjohann_etal:02.pdf.

Goetz, B. (2000). The Lucene search engine: Powerful, flexible, and free. *Java World*, 1-8. Retrieved January 30, 2004 from http://www.javaworld.com/javaworld/jw-09-2000/jw-0915-lucene_p.html.

Gospodnetic, O. (2003). Introduction to Text Indexing with Apache Jakarta Lucene. *OReilly OnJava.com*, 1-7. Retrieved February 13, 2004 from http://www.onjava.com/lpt/a/2944.

Haas, S. (2003). Improving the Search Environment: Informed Decision Making in the Search for Statistical Information. *Journal of the American Society for Information Science and Technology, 54(8)*, 782-797.

Harold, E.R., & Means, W.S. (2002). *XML In a Nutshell*. Sebastopol, California: O'Reilly.

Hatcher, E. (2003). Lucene Intro. *Java.net*, 1-7. Retrieved January 12, 2004 from http://today.java.net/pub/a/today/2003/07/30/LuceneIntro.html.

INEX. (2002). Initiative for the Evaluation of XML Retrieval. Retrieved January 2004 from http://qmir.dcs.qmw.ac.uk/inex/index.html.

Jackson, J., & Gilstrap, D. (1999). XML and better Web searching. *Library Hi Tech, 17(3),* 316-320.

Jakarta Lucene (2003). Retrieved October 2003 from http://jakarta.apache.org/lucene.

Jakarta Lucene FAQ. (2003). Indexing Section. Retrieved January 2004 from http://lucene.sourceforce.net/cgi-bin/faq/faqmanager.cgi?file=chapter.indexing&toc=faq.

Jakarta Lucene FAQ. (2003). Searching Section. Retrieved January 2004 from http://lucene.sourceforce.net/cgi-bin/faq/faqmanager.cgi?file=chapter.search&toc=faq.

Jakarta Lucene Overview. (2003). Query Parser Syntax. Retrieved January 2004 from http://jakarta.apache.org/lucene/docs/.

Jakarta Lucene Resources. (2003). Performance Benchmarks. Retrieved January 2004 from http://jakarta.apache.org/lucne/docs/benchmarks.html.

Kamps, J., Marx, M., de Rijke, M. & Sigurbjornsson, B. (2003). XML Retrieval: What to Retrieve? *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Iinformation Retrieval,* 409-410.

Kotsakis, E. (2002). Structured Information Retrieval in XML Documents. *Proceedings of the ACM SIGIR 2002 Symposium on Applied Computing,* 663-667.

Litkowski, K. (2003a). Question Answering Using XML-Tagged Documents. In E.M. Voorhees & L.P. Buckland (eds.), The Eleventh Text Retrieval Conference (TREC 2002). NIST Special Publication 500-251. Gaithersburg, MD, 122-131. Retrieved January 2004 from http://www.clres.com/online-papers/trec11.pdf.

Liotta, M., & Preimesberger, C. (2003). Native XML databases resolve XML document retrieval issues. *Builder.com*. Retrieved December 2003 from http://www.zdnet.com.au/builder/architect/database/story/0,2000034918,20272675,00.htm.

Losee, R. M. (1998). *Text Retrieval and Filtering: Analytic Models of Performance.* Boston: Kluwer Academic Publishers.

Luk, R., Leong, H.V., Dillon, T. Chan, A. Croft, B. & Allan, J. (2002). A Survey in Indexing and Searching XML Documents. *Journal of the American Society for Information Science and Technology, 53(6),* 415-437.

Mable, G. (2002). The Next Generation Database – XDB. *XML Journal*, 4(6). Retrieved October 2003 from http://www.syscon.com/xml/articleprint.cfm?id=421.

Marchionini, G., Dwiggins, S., Katz, A. Lin, X. (1993). Information seeking in full-text end-user-oriented search systems: The roles of domain and search expertise. *Library & Information Science Research*, 15, 35-69.

Marchionini, G. (1995). *Information Seeking in Electronic Environments*. New York: Cambridge University Press.

Meier, W. (2002). eXist: An Open Source Native XML Database. In Lecture Notes In Computer Science, Revised Papers from the NODe 2002 Web and Database-Related Workshops on Web, Web-Serivces and Database Systems. London: Springer-Verlag, 169-183. Available online at http://exist-db.org/webdb.pdf.

Mertz, D. (2001). XML Matters: Indexing XML Documents. *IBM Developer Works.com.* Retrieved January 2004 from http://www-106.ibm.com/developerworks/xml/library/x-matters10.html.

Mertz, D. (2003). XML Matters: TEI – The Text Encoding Initiative. *IBM Developer Works.com.* Retrieved January 2004 from http://www-106.ibm.com/developerworks/xml/library/x-matters30.html.

Mignet, L., Barbosa, D., Veltri, P. (2003). The XML Web: a First Study. *Proceedings of the 12th international conference on the World Wide Web,* 500-510.

Miller, D. R. (2000). XML Libraries' Strategic Opportunities. *Library Journal, 125(10),* 8-22.

Myaeng, S. H., Jang D., Kim, M., Zhoo, Z. (1998). A flexible model for retrieval of SGML documents. *Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval*, 138-145.

Nambiar, U., Lacroix, Z., Bressan S., Lee, M., Yingguang L. (2002). Current Approaches to XML Management. *IEEE Internet Computing, 6(4),* 43-51.

National Institute of Standards and Technology (NIST). (2000). Retrieved January 2004 from http://www.nist.gov/.

Navarro, G., & Baeza-Yates, R. (1995). A Language for Queries on Structure and Content of Textual Databases. *Proceedings of the 18th Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval, 93-101.

Ogbuji, U. (2001). Thinking XML: XML meets semantics, Part I. Retrieved January 2004 from http://www-106.ibm.com/developerworks/xml/library/x-think1.html.

Oracle. (2004). XML Technology Center. Retrieved December 2003 from http://otn.oracle.com/tech/xml/index.html.

SMART Tutorial. (n.d.). Stopword List (english.stop). Retrieved February 2004 from ftp://ftp.cs.cornell.edu/pub/smart/.

Salminen, A., & Tompa, F. (2001). Requirements for XML Document Database Systems. *Proceedings of the 2001 ACM Symposium on Document Engineering*, 89-94.

Salton, G., & McGill, M.J. (Eds.). (1983). *Introduction to Modern Information Retrieval.* New York: McGraw-Hill.

Shah, U., Finin T., Joshi, A., Cost, R.S., & Mayfield, J. (2003). Information Retrieval on the Semantic Web. *In 10th International Conference on Information and Knowledge Management,* 461-468.

Sleepycat Software.(2003). Berkeley DB. Retrieved November 2003 from http://www.sleepycat.com.

Software AG. (2003). Tamino XML Server. Retrieved January 2004 from http://www.softwareag.com/tamino/.

Southern Folklife Collection. (2004). Retrieved December 2003 from http://www.lib.unc.edu/mss/sfc1/sinv.html.

Suciu, D. (2001). On Database Theory and XML. *SIGMOD Record, 30(3),* 39-45.

Sullivan, D. (1997). The New Meta Tags Are Coming – Or Are They? From The Search Engine Report, Retrieved January 2004 from http://www.searchenginewatch.com/sereport/print.php/34721_2165781.

Tague-Sutcliffe, J. (1992). The Pragmatics of Information Retrieval Experimentation, Revisited. *Information Processing and Management, 28(4),* 467-490.

Tennant, R. (Ed.). (2002). *XML in Libraries*. New York: Neal-Schuman Publishers.

Text Encoding Initiative (TEI).(2001). Retrieved October 2003 from http://www.tei-c.org/.

Text REtrieval Conference (TREC). (2000). Retrieved January 2004 from http://trec.nist.gov/.

Timber. (2003). Retrieved March 2003 from http://www.eecs.umich.edu/db/timber/.

Weiss, S. (1997). Glossary for Information Retrieval. Retrieved November 2003 from http://www.cs.jhu.edu/~weiss/glossary.html.

Wolff, J.E., Florke, H., & Cremers, A.B. (2000). Searching and browsing collections of structural information. *In Proceedings of IEEE advances in digital libraries*, 141-150.

Wilkinson, R. (1994). Effective Retrieval of Structured Documents. *In Proceedings of the Seventeenth Annual International ACMSIGIR Conference on Research and Development in Information Retrieval,* 311-317.

Williams, K. (2001). XML for Data: Native XML databases: a bad idea for data? Retrieved January 2004 from http://www-106.ibm.com/developerworks/xml/library/x-xdnat.html.

World Wide Web Consortium (W3C). (1994). Retrieved December 2003 from http://www.w3.org.

World Wide Web Consortium (W3C) Schools. (1999). XML Syntax. Retrieved December 2003 from http://www.w3schools.com/xml/xml_syntax.asp.

World Wide Web Consortium (W3C) Schools. (1999). XML Validation. Retrieved December 2003 from http://www.w3schools.com/xml/xml_dtd.asp.

World Wide Web Consortium (W3C). (n.d.). On SGML and HTML. Retrieved February 8, 2004 from http://www.w3.org/TR/REC-html40/intro/sgmltut.html.

The Works of Charles Dickens. (2003). Retrieved August 2003 from http://www.ibiblio.org/dickens.

Van Rijsbergen, C. J. (1979). *Information Retrieval* (2nd ed.). Boston: Butterworths.

XYZfind.com.(2004). Retrieved January 2004 from http://www.xyzfind.com/.

Appendix A: Document Collection

File ID	Title of Work
41824.xml	The Posthumous Papers of the Pickwick Club
41826.xml	The New and Elegant Parlour Game of Oliver Twist : for Any Number of Players ... [Game]
41830.xml	The Readings of Mr. Charles Dickens, as Condensed by Himself. David Copperfield and Boots at the HTree Inn.
41832.xml	To Be Read at Dusk [Wise Forgery]
41836.xml	Great Expectations
41837.xml	Sketches of Young Gentleman. Dedicated to the Young Ladies.
41838.xml	Sketches of Young Couples; with an Urgent Remonstrance to the Gentlemen of England (Being Bachelors or Widowers), on the Present Alarming Crisis
41838.xml	Sketches of Young Ladies: in which These Interesting Members of the Animal Kingdom Are Classified, according to Their Several Instincts, Habits, and General Characteristics
41840.xml	A Christmas Carol. In Prose. Being a Ghost Story of Christmas.
41841.xml	The Battle of Life. A Love Story
41842.xml	The Chimes: A Goblin Story of Some Bells That Rang An Old Year Out and a New Year In
41844.xml	The Village of Coquettes: A Comic Opera
41845.xml	Is She His Wife? or, Something Singular. A Comic Burletta in One Act.
41846.xml	The Lamplighter, a Farce. Now First Printed from a Manuscript in the Forster Collection at the South Kensington Museum.
41847.xml	The Frozen Deep. A Drama. In Three Acts.
41848.xml	No Thoroughfare, a Drama in Five Acts.
41849.xml	A Child's Dream of a Star
41851.xml	A Child's Dream of a Star
41852.xml	The Readings of Mr. Charles Dickens, as Condensed by Himself. A Christmas Carol and The Trial from Pickwick.
41853.xml	Dickens Portfolio. Deluxe Edition.
41854.xml	To Be Read at Dusk. in The Keepsake, 1852
42013.xml	Oliver Twist; or, The Parish Boy's Progress, By Boz. In Three Volumes. Vol. I
42014.xml	Oliver Twist; or, The Parish Boy's Progress, By Boz. In Three Volumes. Vol. II
42015.xml	Oliver Twist; or, The Parish Boy's Progress, By Boz. In Three Volumes. Vol. III
42016.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. I
42017.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. II
42018.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. III
42019.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. IV
42020.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. V
42021.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. VI
42022.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. VII
42023.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. VIII
42024.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. IX
42025.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. X
42026.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XI
42027.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the

File ID	Title of Work
	Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XII
42028.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XIII
42029.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XIV
42030.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XV
42031.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XVI
42032.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XVII
42033.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XVIII
42034.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield the Younger. Of Blunderstone Rookery. (Which He Never Meant to Be Published on Any Account.) No. XIX & XX
42035.xml	A Tale of Two Cities, Vol. I
42036.xml	A Tale of Two Cities, Vol. II
42037.xml	A Tale of Two Cities, Vol. III
42038.xml	A Tale of Two Cities, Vol. IV
42039.xml	A Tale of Two Cities, Vol. V
42040.xml	A Tale of Two Cities, Vol. VI
42041.xml	A Tale of Two Cities, Vol. VII & VIII
42042.xml	Great Expectations. In Three Volumes. Vol. I
42043.xml	Great Expectations. In Three Volumes. Vol. II
42044.xml	Great Expectations. In Three Volumes. Vol. III
42045.xml	Bleak House, Vol. I
42046.xml	Bleak House, Vol. II
42047.xml	Bleak House, Vol. III
42048.xml	Bleak House, Vol. IV
42049.xml	Bleak House, Vol. V
42050.xml	Bleak House, Vol. VI
42051.xml	Bleak House, Vol. VII
42052.xml	Bleak House, Vol. VIII
42053.xml	Bleak House, Vol. IX
42054.xml	Bleak House, Vol. X
42055.xml	Bleak House, Vol. XI
42056.xml	Bleak House, Vol. XII
42057.xml	Bleak House, Vol. XIII
42058.xml	Bleak House, Vol. XIV
42059.xml	Bleak House, Vol. XV
42060.xml	Bleak House, Vol. XVI
42061.xml	Bleak House, Vol. XVII
42062.xml	Bleak House, Vol. XVIII
42063.xml	Bleak House, Vol. XIX & XX
42064.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield, the Younger, of Blunderstone Rookery : (Which He Never Meant to be Published on Any Account) Vol. I
42065.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield, the Younger, of Blunderstone Rookery : (Which He Never Meant to be Published on Any Account) Vol. II
42066.xml	The Personal History, Adventures, Experience, and Observation of David Copperfield, the Younger, of Blunderstone Rookery : (Which He Never Meant to be Published on Any Account) Vol. III
42067.xml	The Posthumous Papers of the Pickwick Club, Vol. I
42068.xml	The Posthumous Papers of the Pickwick Club, Vol. II
42069.xml	The Posthumous Papers of the Pickwick Club, Vol. III
42070.xml	The Posthumous Papers of the Pickwick Club, Vol. IV
42071.xml	The Posthumous Papers of the Pickwick Club, Vol. V

File ID	Title of Work
42072.xml	The Posthumous Papers of the Pickwick Club, Vol. VI
42073.xml	The Posthumous Papers of the Pickwick Club, Vol. VII
42074.xml	The Posthumous Papers of the Pickwick Club, Vol. VIII
42075.xml	The Posthumous Papers of the Pickwick Club, Vol. IX
42076.xml	The Posthumous Papers of the Pickwick Club, Vol. X
42077.xml	The Posthumous Papers of the Pickwick Club, Vol. XI
42078.xml	The Posthumous Papers of the Pickwick Club, Vol. XII
42079.xml	The Posthumous Papers of the Pickwick Club, Vol. XIII
42080.xml	The Posthumous Papers of the Pickwick Club, Vol. XIV
42081.xml	The Posthumous Papers of the Pickwick Club, Vol. XV
42082.xml	The Posthumous Papers of the Pickwick Club, Vol. XVI
42083.xml	The Posthumous Papers of the Pickwick Club, Vol. XVII
42084.xml	The Posthumous Papers of the Pickwick Club, Vol. XVIII
42085.xml	The Posthumous Papers of the Pickwick Club, Vol. XIX & XX
41816.xml	The Cricket on the Hearth : A Fairy Tale of Home

Appendix B: Natural Language Queries and their Translations

		Query Type			
		XPath		Keyword	
Query	**Natural Language**	**Simple**	**Complex**	**Simple**	**Complex**
Q1	How many of the works in the collection are inscribed or dedicated to someone Dickens knew?	//TEI.2 |="inscribed dedication"	//div1[@type="dedication"]	inscribed OR dedication	contents: inscribed OR contents: dedication
Q2	Are there any Diamond Editions in the collection?	//TEI.2[near(., 'diamond edition')]	//biblFull/titleStmt/title[near(., 'diamond edition')]	"diamond edition"	contents: "diamond edition"
Q3	Are there any portfolios in the collection?	//TEI.2 &="portfolio"	//biblFull/titleStmt &="portfolio"	portfolio	title: portfolio
Q4	Which edition(s) of A Christmas Carol does the library own?	//TEI.2[near(., 'christmas carol')]	//biblFull/titleStmt[near(., 'christmas carol')]	"christmas carol"	title: "christmas carol"
Q5	Which works were published by Ticknor?	//TEI.2 &="ticknor"	//biblFull/publicationStmt/publisher &="ticknor"	ticknor	publisher: ticknor
Q6	Are any of the works in the collection limited editions?	//TEI.2[near(., 'limited edition')]	//biblFull/publicationStmt/publisher[near(., 'limited edition')]	"limited edition"	publisher:"limited edition"
Q7	Which works were published in London after 1867?	//TEI.2 &= "london 1867"	//biblFull/publicationStmt[pubPlace &="london" and date>=1867]	london AND 1867	pubPlace: london AND pubDate: [1867 TO 2000]
Q8	Are any of the works in the collection copyrighted editions?	//TEI.2[near(., 'copyright edition', 0)]	//edition &="copyright"	"copyright edition"	edition: copyright
Q9	Which works in the collection were engraved by Dalziel?	//TEI.2 &="dalziel"	//div1[@type &="illustration*"]/p &= "dalziel"	dalziel	illustrator: daizel
Q10	Which works have illustrations by Richard Doyle?	//TEI.2 &="doyle"	//div1[@type &="illustration*"]/p &= "doyle"	doyle	illustrator: doyle
Q11	What is the Rare Books Library call# of the Cricket on the Hearth? A Fairy Tale of Home	//TEI.2[near(., 'cricket hearth')]	//biblFull[titleStmt &="cricket hearth" and notesStmt]	"cricket hearth"	title: "cricket hearth"
Q12	Who published The Village Coquettes?	//TEI.2[near(., 'village coquettes')]	//biblFull[titleStmt &="village coquettes" and publicationStmt]	"village coquettes"	title: "village coquettes"
Q13	Are there any references to elections in any of the works in the collection, either in advertisements, books or sketches?	//TEI.2 &="election"	//text[body &="election"]	election	bookText: election
Q14	Are there any works that mention railroads or railways?	//TEI.2 |="railroad railway"	//text[body &="railroad" or body &="railway"]	railroad OR railway	bookText: railroad OR bookText: railway
Q15	Does Dickens mention slavery in any of the works included in the collection?	//TEI.2 &="slavery"	//text/body &="slavery"	slavery	bookText: slavery

Appendix C: Relevance Judgments

	Query															
Doc ID	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**	**TOTAL**
41816	x								x	x	x					**4**
41824		x			x								x	x	x	**5**
41830		x			x			x						x		**4**
41836													x		x	**2**
41837	x												x			**2**
41840				x												**1**
41841	x								x	x						**3**
41842									x	x			x			**3**
41844	x											x				**2**
41845					x			x								**2**
41846							x									**1**
41848							x							x		**2**
41851							x									**1**
41852		x		x	x			x								**4**
41853			x			x										**2**
42014													x			**1**
42016														x		**1**
42017														x		**1**
42018														x		**1**
42019														x		**1**
42024															x	**1**
42026														x		**1**
42027														x		**1**
42029														x		**1**
42030														x		**1**
42031														x		**1**
42033														x	x	**2**
42034	x													x		**2**
42036														x		**1**
42037															x	**1**

	Query															
Doc ID	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**	**TOTAL**
42041	x															**1**
42042	x												x			**2**
42043													x		x	**2**
42047													x			**1**
42048													x			**1**
42050													x			**1**
42052													x			**1**
42057													x			**1**
42059														x		**1**
42060													x			**1**
42061														x		**1**
42063	x													x		**2**
42064								x								**1**
42065								x							x	**2**
42066	x							x							x	**3**
42069														x	x	**2**
42070													x			**1**
42071													x			**1**
42072													x			**1**
42073													x			**1**
42083													x			**1**
42084													x			**1**
42085	x												x			**2**
TOTAL	**10**	**3**	**1**	**2**	**4**	**1**	**3**	**6**	**3**	**3**	**1**	**1**	**20**	**19**	**9**	**86**

Appendix D: System and Index Configuration

Variable Description	***Variable Value***
Hardware Environment	
Dedicated Machine for Indexing	No, indexing is stored on machine with other programs and the Exist and Lucene installations
CPU	Pentium 4, 2.4 GHz single processor
RAM	512 MB
Harddrive Size	80 GB
Network	100 Base T Network
Software Environment	
Java Version	J2SDK 1.4.2
Java VM	Server/client VM
OS Version	Linux RedHat 9
Exist Version	eXist 0.9.2
Lucene Version	Lucene 1.3
Location of index for Lucene	/share/lucene_index
Location of index for Exist	/share/exist_index
Other significant software	mySQL database; Apache Http proxy server; Jetty Servlet Container (comes with Exist and Lucene), eclipse SDK 2.1.2 (Integrated Development Environment)
Indexing Variables	
Number of source documents	95
Total filesize of source documents	21.55 MB
Average filesize of source documents	0.23 MB
Average Nodes in source documents	2050
Average Attributes in source documents	3939
Location of source documents	/share/dickens/xml
File type of source documents	XML/TEI encoded files
Parser(s) used (Lucene only)	default Query parser
Analyzer(s) used (Lucene only)	StandardAnalyzer class, using stop words as the constructor argument

Appendix E: Test I & II Results at the Query Level

[1] Rows are highlighted where 100% precision and recall were attained.

[2] Query results are aggregated in Appendix F.

Exist*

	QUERY TYPE			
	SIMPLE		COMPLEX	
Query ID	RECALL	PRECISION	RECALL	PRECISION
1	1.00	0.42	0.90	1.00
2	0.67	0.67	0.00	0.00
3	1.00	0.08	1.00	1.00
4	1.00	0.08	1.00	1.00
5	1.00	0.80	1.00	1.00
6	1.00	1.00	1.00	1.00
7	0.00	0.00	1.00	1.00
8	0.67	0.80	0.67	1.00
9	1.00	0.43	1.00	1.00
10	1.00	0.10	1.00	1.00
11	1.00	1.00	1.00	1.00
12	1.00	1.00	1.00	1.00
13	0.95	0.90	0.80	1.00
14	0.95	0.43	0.58	1.00
15	1.00	0.53	1.00	1.00

* Statistics were the same for both test runs and thus were compiled in 1 table

Lucene Test I

	QUERY TYPE			
	SIMPLE		COMPLEX	
Query ID	RECALL	PRECISION	RECALL	PRECISION
1	0.70	0.39	0.00	0.00
2	0.67	1.00	0.67	1.00
3	1.00	0.17	0.00	0.00
4	1.00	0.08	0.00	0.00
5	1.00	0.80	0.00	0.00
6	1.00	1.00	0.00	0.00
7	0.33	0.25	0.00	0.00
8	0.83	0.83	0.00	0.00
9	1.00	0.75	0.00	0.00
10	1.00	0.12	0.00	0.00
11	1.00	0.05	0.00	0.00
12	1.00	1.00	0.00	0.00
13	0.65	1.00	0.00	0.00
14	0.89	0.53	0.00	0.00
15	0.44	0.57	0.00	0.00

Lucene Test II

	QUERY TYPE			
	SIMPLE		*COMPLEX*	
Query ID	*RECALL*	*PRECISION*	*RECALL*	*PRECISION*
1	0.70	0.39	0.70	0.39
2	0.67	1.00	0.67	1.00
3	1.00	0.17	1.00	1.00
4	1.00	0.08	1.00	1.00
5	1.00	0.80	1.00	1.00
6	1.00	1.00	1.00	1.00
7	0.33	0.25	1.00	1.00
8	0.83	0.83	0.67	1.00
9	1.00	0.75	0.00	0.00
10	1.00	0.12	1.00	1.00
11	1.00	0.05	1.00	1.00
12	1.00	1.00	1.00	1.00
13	0.65	1.00	0.70	0.93
14	0.89	0.53	0.95	0.56
15	0.44	0.57	0.44	0.50

Appendix F: Descriptive Statistics for Test I and II

Descriptive statistics for both systems outline how simple and complex queries performed.

Exist*

	QUERY TYPE			
	SIMPLE		COMPLEX	
	PREC	RECALL	PREC	RECALL
Mean	.5493	.8827	.9333	.8633
Median	.5300	1.0000	1.0000	1.0000
Mode	1.00	1.00	1.00	1.00
Std. Deviation	.36472	.26943	.25820	.27453
Variance	.13302	.07259	.06667	.07537
Range	1.00	1.00	1.00	1.00
Minimum	.00	.00	.00	.00
Maximum	1.00	1.00	1.00	1.00

* Statistics were the same for both test runs and thus were compiled in 1 table

Lucene Test I

	QUERY TYPE			
	SIMPLE		COMPLEX	
	PREC	RECALL	PREC	RECALL
Mean	.5693	.8340	.0667	.0447
Median	.5700	1.0000	.0000	.0000
Mode	1.00	1.00	.00	.00
Std. Deviation	.36858	.22545	.25820	.17299
Variance	.13585	.05083	.06667	.02993
Range	.95	.67	1.00	.67
Minimum	.05	.33	.00	.00
Maximum	1.00	1.00	1.00	.67

Lucene Test II

	QUERY TYPE			
	SIMPLE		COMPLEX	
	PREC	RECALL	PREC	RECALL
Mean	.5693	.8340	.8253	.8087
Median	.5700	1.0000	1.0000	1.0000
Mode	1.00	1.00	1.00	1.00
Std. Deviation	.36858	.22545	.31202	.28844
Variance	.13585	.05083	.09736	.08320
Range	.95	.67	1.00	1.00
Minimum	.05	.33	.00	.00
Maximum	1.00	1.00	1.00	1.00