THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Doctor of Philosophy (Computer Science)

DEGREE

Computer Science	Computer Science
FIELD	**DEPARTMENT**

TITLE: Optimizing XML Element Retrieval Using Compression and Double Scoring Techniques

NAME: Mr. Tanakorn Wichaiwong

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR

(Associate Professor Chuleerat Jaruskulchai, D.Sc.)

ACTING DEPARTMENT HEAD

(Mr. Chakrit Watcharopas, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN

(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

OPTIMIZING XML ELEMENT RETRIEVAL USING COMPRESSION AND DOUBLE SCORING TECHNIQUES

TANAKORN WICHAIWONG

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy (Computer Science)
Graduate School, Kasetsart University
2013

Tanakorn Wichaiwong 2013: Optimizing XML Element Retrieval Using Compression and Double Scoring Techniques. Doctor of Philosophy (Computer Science), Major Field: Computer Science, Department of Computer Science. Thesis Advisor: Associate Professor Chuleerat Jaruskulchai, D.Sc. 122 pages.

The content and structure properties of XML allow focused access to documents using XML retrieval systems. This enables the return of components of documents, rather than the whole document, in response to a user query. Current research on XML retrieval has attempted to solve two main issues in targeted information retrieval: the problem of the smallest unit of information to be retrieved and the problem of nested element. The selected type approach is one such XML ranking strategy that utilizes a selected element list and ranks the elements most relevant to users. However, this strategy is problematic if all potentially retrievable elements need to be manually chosen. In order to solve this issue, in this thesis we introduce two automatic methods that enable a) tuning weights of fields using mixed content elements in document-centric XML, and b) assignment of weights to the selected fields, called the Double Scoring algorithm.

Furthermore, to determine the appropriate level of document components, we propose a novel technique that assigns an element score to each component by sharing scores from leaf nodes to their parents, which we call the Score Sharing algorithm. We utilize this algorithm to improve the effectiveness of a search system in terms of the iP[0.01] and MAiP; the improvements are 26.82% and 40.14%, respectively. In addition, we propose a new XML compression algorithm called the extension XML compression of ADXPI. These steps reduce the size of the data by 90.19%, when compared to GPX, and in addition, reduce the query processing time by 37.12% compared to that before compression.

_____ _____ ___/___/___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Associate Professor Chuleerat Jaruskulchai, my thesis advisor, for advice, encouragement, and valuable comments for my thesis. It was she who first introduced me to the field of information retrieval, and she taught me a great deal about research methodology. I also would like to thank Professor Jacek Kitowski and Dr. Miro Lehtonen for their valuable comments and suggestions and their patience in correcting my English. I would sincerely like to thank the other members of my committee– Professor Vilas Wuwongse, Associate Professor Thanaruk Theeramunkong, Assistant Professor Arnon Rungsawang, Professor Chidchanok Lursinsap, Associate Professor Peraphon Sophatsathit, Associate Professor Nuanwan Soonthornphisaj, Associate Professor Anongnart Srivihok and Assistant Professor Sirikorn Channual who provided useful comments and helped to bring my work into its final form. I give my heartfelt thanks to all of the teachers and officers of the Department of Computer Science.

I owe a great dept of gratitude to my parents for all they have given me and for their continuing encouragement. Finally, I would like to thank my wife, Ms. Angrisa Tawonatiwasna, for all her support and encouragement in my life.

Tanakorn Wichaiwong

May 2013

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF TABLES (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Appendix Figure

LIST OF ABBREVIATIONS

XML	=	eXtensible Mark-up Language
IR	=	Information Retrieval
DTD	=	Document Type Definition
MEXIR	=	More Efficient XML Information Retrieval
TF	=	Term Frequency
IDF	=	Inverse Document Frequency
DL	=	Document Length
HTML	=	Hypertext Markup Language
CO	=	Content Only
CAS	=	Content and Structure
NEXI	=	The Narrowed Extended XPath I
Flex	=	Flexible Retrieval
GPX	=	Garden Point XML Retrieval
BUS	=	Bottom-Up Scheme
TRIX	=	Tampere information retrieval and indexing of XML
RDF	=	Resource Description Framework
nxCG	=	normalized extended Cumulated Gain
CG	=	Cumulated Gain
AiP	=	Average interpolated Precision
MAiP	=	Mean Average interpolate Precision
WSDL	=	Web Services Description Language
AutoMix	=	Automatic Tuning of Mixed Element
SW	=	Selected Weight
ADXPI	=	Absolute Document XPath Indexing
DC	=	Data Centric

OPTIMIZING XML ELEMENT RETRIEVAL USING COMPRESSION AND DOUBLE SCORING TECHNIQUES

INTRODUCTION

Nowadays, the wide use of eXtensible Mark-up Language (XML) documents in digital libraries has led to the development of Information Retrieval (IR) methods specifically designed for XML collections. XML Documents have both a content and a structure; the XML content refers to the text within the document, and the XML structure refers to the logical organization of the document. Since XML documents separate content and structure, XML-IR systems are able to retrieve relevant portions of documents.

An XML retrieval system always presents the most specific part of a document, instead of the whole document, and also determines the appropriate level of element granularity to return to users. The granularity of an element depends on the maximum coverage and density of the query terms, and also a location within the XML document where users should start reading the content that is relevant to their information needs (Kamps *et al.*, 2007a). For instance, in response to a user query on a Wikipedia collection, XML retrieval systems may return a group of paragraph and section elements that have been estimated to be the best answer. Pharo (Pharo, 2008) discovered in interactive experiments that users prefer to use smaller sections instead of a whole article. The XML retrieval research community has been trying to solve the problem of returning the appropriate element level in answering a query because the queried information lies in the smallest unit of complete information, and at the same time, any nested elements must be reduced. Approaches to XML retrieval systems may be divided into three types, namely, element based, combination based and selected type based. In element based approaches, each element is indexed based on both the direct text and the text of descendants and the output is filtered to remove nested elements. One approach to element scoring is based on a multinomial language model that combines the language models for the element and for the whole collection (Sigurbjörnsson and Kamps, 2005). Crouch (Crouch, 2006) proposed a method which is based on a rank-ordered list of retrieved elements from an all element index and in which the

output is filtered to remove various nested elements. To overcome the nested element issue based on the combination based approach, these approaches are extended by combining the scores of leaf elements and assigning the combined scores to the non-leaf elements (collapsing several nested elements). Geva (Geva, 2007) presented a score propagation algorithm which is a weighted accumulation of ranking scores of an element's children that is assigned to their parent. Score Aggregation is based on the aggregated representation of the term statistics of its own context with the statistics of the element's children (Ogilvie and Callan, 2005). Arvola et al. (Arvola *et al.*, 2011) proposed the Contextualization function, which is a general re-weighting method in which any context of the element along the hierarchical path may influence the weight of the element. The result showed that root contextualization was the best. However, the root might carry large units of retrieval. These approaches have a drawback in terms of response time because retrieving score of the non-leaf elements require considerable time and must be calculated at query time. Selected type approaches try to solve the problem of small units of information and nested elements by using a threshold of element size and the significance of elements. Previous studies have shown what the significant parts of a document are, for not all parts have equal influence (Trotman, 2005). For example, the organization of a journal article is as follows: the "title" contains a few words that concisely describe the topic of the article, the "abstract" gives an outline of the contents in a few paragraphs, the "body" of the article consist of several pages that describe the work precisely, and finally, the "conclusion" summarizes the content. In fact, the terms that appear in some specific document fields or XML elements should have a greater influence than other elements. Consequently, a term that appears in the element "title" might be more important than a term that appears in the "abstract" or "body" elements. These heuristics have been applied successfully to optimize IR systems that use BM25F (Stephen *et al.*, 2004; Stephen and Hugo, 2009) which considers multiple document fields with possibly different degrees of importance called selected fields.

The BM25F function has performed well in past evaluations; however, there are issues that require additional attention. Firstly, which XML elements should be treated as fields? Secondly, what is an appropriate weight for each field? Previously, document fields were selected manually, and the weight for each chosen field was tuned before

being assigned. This exacerbated preprocessing complexity with respect to cost and time for heterogeneous collections (Stephen and Hugo, 2009).

Even though XML structure can be beneficial, it can be time and space consuming, and also influences the efficiency of the search system (Anh and Moffat, 2002; Fuhr and Gövert, 2002). The main disadvantage of using XML documents is their large size due to the fact that are highly structured and often have long tag and attribute names; for instance, the element name in Wikipedia is "management-note[1]", "broadcasting-station[1]", "system-of-measurement[1]". Recent developments in XML compression techniques are aimed at not only reducing XML data storage, but also increasing efficiency. XML data compression techniques can be divided into two types with respect to their ability to support queries: 1) Non-Queriable XML Compressors are aimed to achieve the highest compression ratio and 2) Queriable XML Compressors are aimed to perform direct queries on compressed data. For the queriable compressors, Tolani and Haritsa (Tolani and Haritsa, 2002) proposed a technique called XGrind. However, XGrind will compress only XML data that has a Document Type Definition (DTD) structure. XPRESS (Min *et al.*, 2003) embodies a new idea which uses reverse arithmetic encoding, which is a method for organizing data so that the search for XPath expressions can be done effectively. However, the use of XPRESS is limited because XPRESS cannot understand documents that use ID and IDREF. Wichaiwong and Jaruskulchai (Wichaiwong and Jaruskulchai, 2007) proposed a technique called XSchemaTag that compresses only XML elements and yet permits search and maintenance of documents. However, the XSchemaTag technique does not take into account the frequency of tags and the position of tags (or elements). In XML retrieval systems, the frequency of tag occurrences and the position of tags are very important. It is possible to modify the queriable type compression algorithms to handle this issue.

Contributions of this thesis

This thesis presents novel methods specifically designed for XML retrieval that will improve both efficiency and effectiveness of the search system. The four main contributions of the thesis, which have been presented in previous publications in

international journals and conferences or workshops are as follows:

1. A new algorithm with scoring and returning elements at the right level of granularity called the Score Sharing algorithm (Wichaiwong and Jaruskulchai, 2010 2012d).

2. A new XML scoring algorithm to overcome the issues of the BM25F ranking function called the Double Scoring algorithm (Wichaiwong and Jaruskulchai, 2011e 2012c 2013).

3. A new compression algorithm to optimize the query processing time and reduce storage space (Wichaiwong and Jaruskulchai, 2011b 2012ab).

4. A new XML retrieval system to automate algorithms called the More Efficient XML Information Retrieval (MEXIR) (Wichaiwong and Jaruskulchai, 2011c 2012c).

OBJECTIVES

1. To optimize the XML scoring method so as to return the appropriate level of component granularity by developing a new XML scoring algorithm.

2. To overcome the research issues of the selected type approach by using the mixed content of an XML document and automatically assigning the weight.

3. To optimize query processing time and reduce storage space by developing a new compression algorithm.

4. To implement a new prototype XML retrieval system.

Thesis Outline

This thesis is organized as follows: related works are reviewed in the literature review section, which gives an overview of XML-IR, XML query languages and ranking methods, and compression techniques in relation to XML retrieval. In the methods section, novel methods specifically designed for XML retrieval are proposed. The first is a new algorithm which scores and returns elements at the right level of granularity. The second is a new algorithm to overcome the research issues of the selected type approach. In addition, this thesis develops a new compression algorithm to optimize query processing time. The experiments that were performed on these algorithms and that validated the automated procedures are explained in the results and discussion section, and the conclusions and recommendations are presented at the end of this thesis.

LITERATURE REVIEW

In this chapter, some historical perspectives are provided on areas of research that have influenced basic retrieval and search engine tools related to this thesis as follows: XML data models are explained in Section 1. Classical information retrieval is described in Section 2. Previous research on information retrieval on XML is described in Section 3. XML ranking methods are described in Section 4. XML evaluation methods are explained in Section 5. XML compression techniques of XML are explained in Section 6.

1. XML Data Models

The basic data model of XML (Bray *et al.*, 2008) is a labelled and ordered tree. Figure 1 shows the data tree of a XML document based on the node-labelled model. There are basically three types of nodes in a data tree;

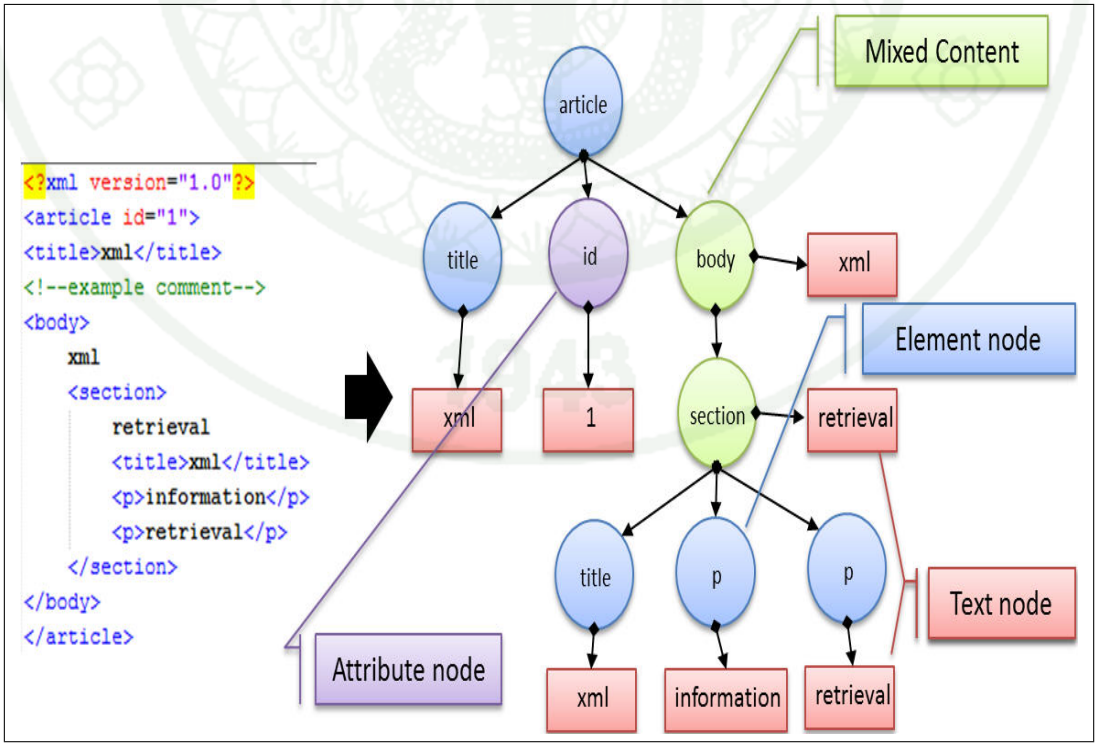


Figure 1 An Example of an XML Element Tree

1. Element nodes correspond to tags in XML documents and may contain one or more elements, the boundaries of which are delimited by begin and end tags, for instance, “title” and “p” nodes.

2. Attribute nodes correspond to attributes associated with tags in XML documents. In contrast to element nodes, attribute nodes are not nested (that is, an attribute cannot have any sub elements), and are unordered (that is, attributes of an element can freely interchange their occurrence locations under the element), for instance, the “id” node.

3. Text nodes correspond to data values in XML documents, for instance, “xml”, “information” and “retrieval” nodes.

4. Mixed content nodes are a type of element that may contain text with child elements, for instance, “body” and “section” nodes.

5. Comment nodes, which are enclosed in $<!--$ and $-->$ are useful for human readers and are ignored by the processor, for instance, “example comment”.

XML documents are commonly divided into two categories, namely data-centric and document-centric or text-centric documents (Manning *et al.*, 2008). Data-centric documents are characterized by a fairly regular structure where the smallest independent unit of data is at the level of an element or an attribute, and there is no *mixed content*. Most data-centric XML documents are stored in databases. Data-centric approaches are concerned with the querying of XML documents from a database with exact matching conditions. In addition, no ranking of results is offered. Examples of data-centric documents are e-Commerce documents, scientific data, and Web services documents.

Document-centric or text-centric documents are usually designed for texts such as books, emails, advertisements, and almost any written document. These documents are characterized by a less regular or an irregular structure, and a large amount of *mixed content*. The order in which elements and context occur is very important to the understanding of the meaning of the document; the text cannot be re-ordered. Unlike data-centric documents, document-centric documents serve mainly to expose the logical

structure of document text. Document-centric approaches extend information retrieval strategies to deal with long text fields, best match conditions (if a document has only one of the query terms, it will be matched), and required ranking results. For example, Figure 2 depicts a document-centric chapter of a book.

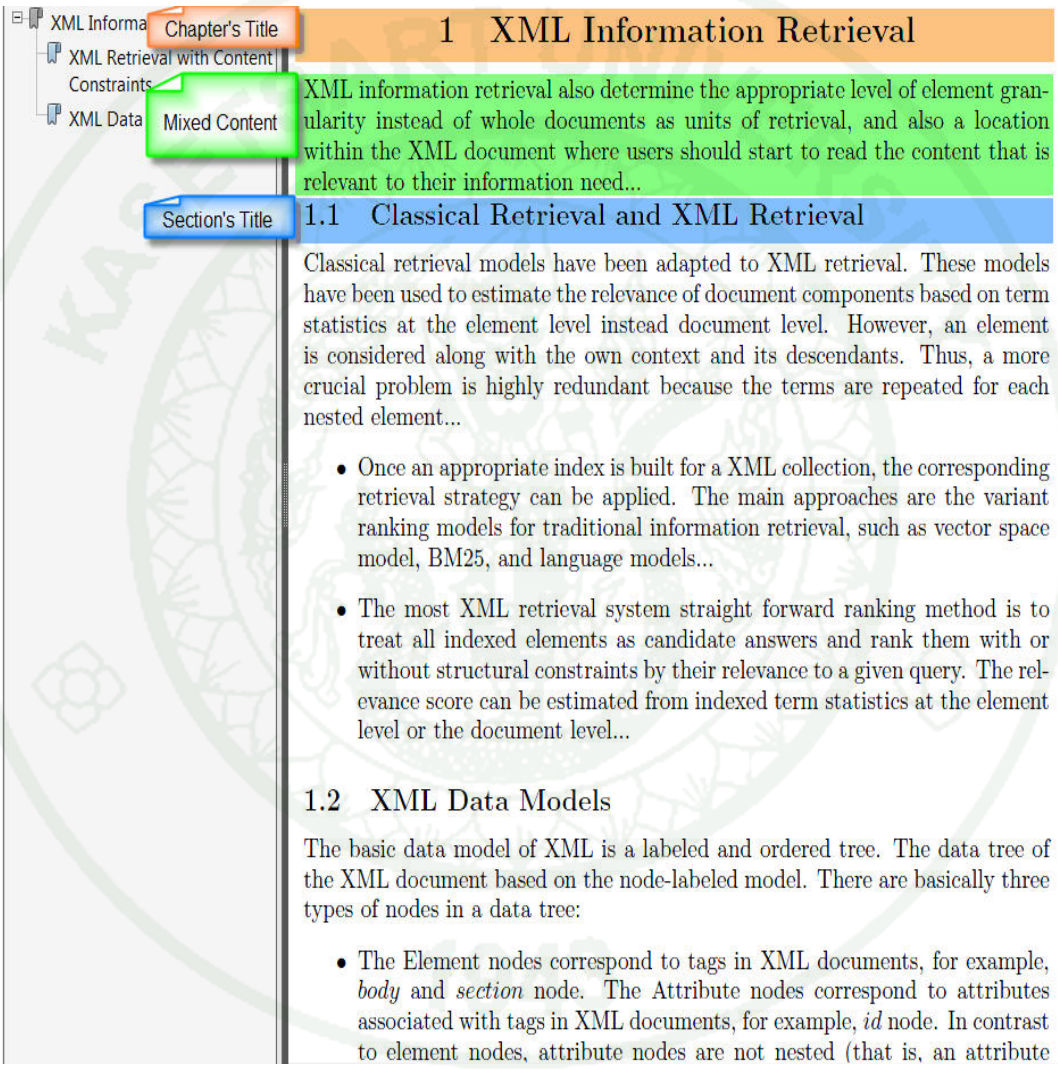


Figure 2 An example of a chapter of a book

Generally, the body of the book contains the description, explanation, argument and/or narrative in full detail; it is the main text of the book. The bodies of longer books are often divided into chapters. A synopsis (summary) of the descendants (chapters) may appear between the Chapter title and the descendants. Without the summary,

it would be difficult for the reader to know what the contents of the chapters are about and it would require more time and effort to read through them (Rzadkiewicz, 2011). Figure 2 shows the summary of a book chapter starting from "XML information retrieval..." in the body of the chapter. Note that, the "Chapter" element contains text, and it is followed by a child element which is "Section Title" (i.e. *mixed content* according to W3C). This provides further information on the content of the chapter and its descendants. In our view, the summary and *mixed content* are of critical importance to understanding the descendants. As a result, the terms that appear in the *mixed content* are necessary and can be used to identify highly relevant elements.

2. Classical IR Ranking Methods

Before discussing ranking strategies of XML retrieval, it is helpful to first obtain an understanding of classical IR ranking methods and the basic factors involved. Generally, the three basic factors in considering the importance of a term in a context is the term's frequency, its inverse document frequency and the document length. These factors may be briefly explained as follows:

1. The Term Frequency (TF) Factor is based on the notion that the importance of a term increases with the term statistics in the element or document context. This factor uses the number of occurrences of a term to estimate the weight assigned to it.

2. The Inverse Document Frequency (IDF) Factor is based on the consideration that common words such as "the" appear in many articles and have very high term frequencies. On the other hand, uncommon words such as "retrieval" appear in few articles and have very low term frequencies. A match between a query and an article on a word with a specific meaning makes the article more relevant to a user in contrast to a match on a common word. For this reason, this factor has to assign different weights to terms in context.

3. Document Length (DL) Factor is based on the consideration that documents vary greatly in length and that longer documents have higher term frequencies and more unique terms. In long documents, the same term may appear repeatedly and numerous different words may be used; as a result, the term frequencies of longer documents are

higher and more terms match a user query. For this reason, we have to compensate for the length and remove the bias in favour of long documents.

Information retrieval systems assign a numeric score to every document and rank documents by this score. Several models have been proposed for this process. The two most-used models in IR research are the vector space model and the probabilistic model. The three systems with the best base performance are Smart (Salton, 1965), Okapi BM25 (Stephen *et al.*, 1994) and INQUERY (Broglio *et al.*, 1994a). These are term weight systems developed for unstructured information retrieval. The Smart system is a text processing system based on a vector space model that was developed over thirty years ago. The Okapi and INQUERY systems use a probabilistic based technique to obtain the document and the query term weights.

In the vector space model (Salton, 1965) the information is represented by a vector of terms. If words or phrases are chosen as terms, then the weight for each term is based on the term's occurrences in a given document. The query is also modelled the same as the document vector, and a matching function is used to evaluate for each document and query as follows:

$$Score(d,q) = \sum_{t \in q} tf_t * idf_t \tag{1}$$

$$idf_t = \log\left[\frac{N}{d_t}\right] \tag{2}$$

where

$Score(d,q)$ measures the relevance of document d to a query q.

tf_t is the frequency of a term t occurring in an document d.

N is the total number of documents in the collection.

d_t is the number of total documents in which the term t occurs.

The probabilistic model, Okapi BM25 (Stephen *et al.*, 1994) is based on the general principle that documents in a collection should be ranked in order of decreasing probability of their relevance to a query. These models weight a term in a document on the basic of the probability that the term will appear in relevant and non relevant documents as follows:

$$Score(d,q) = \sum_{t \in q} \frac{(k_1+1)*tf_t}{K+tf_t} * \log\left[\frac{N-d_t}{d_t}\right] * \frac{(k_3+1)*tf_{t,q}}{k_3+tf_{t,q}} \quad (3)$$

$$K = k_1 * \left((1-b)+b*\frac{len(d)}{avdl}\right) \quad (4)$$

where

$Score(d,q)$ measures the relevance of document d to query q.

tf is the frequency of term t occurring in document d.

$len(d)$ is the length of document d.

$avdl$ is the average length of document in the entire collection.

k_1, k_3 and b are used to balance the weight of term frequency and document length.

The INQUERY system (Broglio *et al.*, 1994a) like most statistical systems, relies on a TF-IDF formula for estimating the probability of a document's relevance to a query as follows:

$$Score(d,q) = d_b + (1-d_b) * \left(tf_m * H + (1-tf_m) * \log\left[\frac{tf+0.5}{max(tf)+1}\right]\right) * idf \quad (5)$$

where

$Score(d,q)$ measures the relevance of document d to query q.

tf is the frequency of term t occurring in document d.

$max(tf)$ is the frequency of the most frequent term in the document d.
tf_m is the minimum term frequency component when a term occurs in a document d.
d_b is the minimum belief component when a term occurs in a document d.
H is a smoothing parameter (according to (Broglio *et al.*, 1994b)).

$$H = \begin{cases} 1.0 & ; \text{if } max(tf) \leq 200 \\ 200 \text{ / max(tf)} & ; \text{otherwise} \end{cases} \quad (6)$$

3. Information Retrieval On XML

Generally, textual information can be broadly classified into two categories: unstructured and structured texts. These may be described as follows: An unstructured text has no fixed predefined format, and is typically expressed in natural language. For example, the information available on the web is unstructured. Although this information is mostly in the Hypertext Markup Language (HTML) format, thus imposing some structure on the text, the structure is only for the purpose of presentation. A structured text, on the other hand, is usually represented in XML format, with mark-up tags that are not involved in the presentation but are used to describe the stored content.

Classical IR focuses on identifying documents that are estimated to be relevant to an information need, typically expressed by the user in the form of a query as shown in Figure 3. The granularity of answers in this case is fixed and whole documents are usually returned to the user's request (Baeza-Yates and Ribeiro-Neto, 1999). In contrast, XML information retrieval always presents the most specific part of a document as a unit of retrieval, instead of the whole document (Amer-Yahia and Lalmas, 2006; Manning *et al.*, 2008). Moreover, XML retrieval also seeks out the location within an XML document where the content is relevant to users' queries and where users should start to read (Kamps *et al.*, 2007a). The query usually consists of a set of terms, optionally including phrases or logical query operators. Classical IR query languages usually express only keyword terms, and do not include any structure. On the other

hand, XML-IR's query languages may add content and structures. A brief summary is presented in the next section.

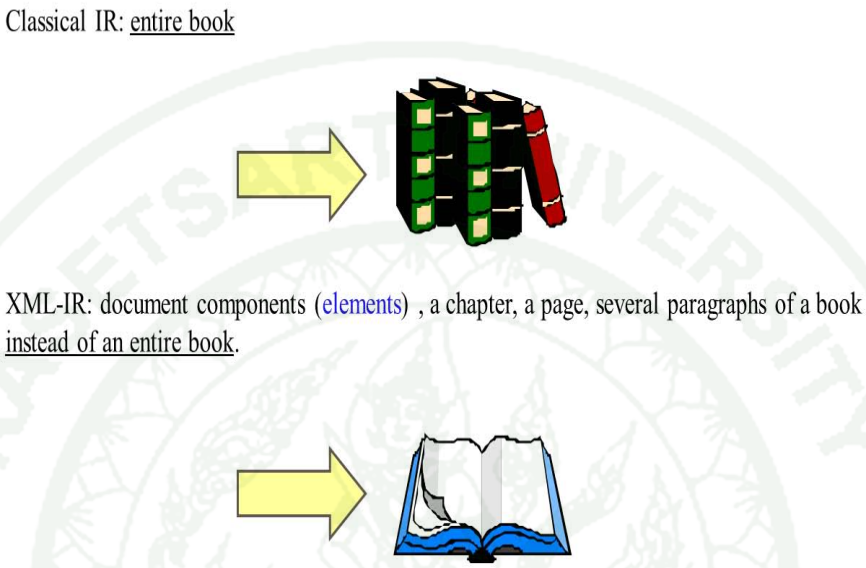


Figure 3 The unit of retrieval in Classical IR and XML-IR

3.1 Querying with XML Document

Querying in structured documents must be with respect to content and structure. INEX identified two types of queries (Trotman and Sigurbjörnsson, 2004b; Trotman and Lalmas, 2005) they are Content Only (CO) and Content and Structure (CAS) as follows:

3.1.1 Content Only Queries

These queries are formed by ignoring the document structure, in the same way as the traditional queries used in IR collections. However, they pose a challenge to XML retrieval in that the retrieval results in returning document components, i.e. XML elements instead of whole documents in response to a user query. Queries can be elements of various complexities, i.e. at different levels of the XML document's structure. This is suitable for XML retrieval where users do not

know, or are not concerned about the structure, i.e., with the logical organization of the document, when expressing their information needs. For example, the best answer for a query "XML retrieval" applied to Figure 1 may be a "section" and not "title" or "p" elements.

3.1.2 Content and Structure Queries

These queries contain conditions of both content and structure. These conditions may refer to the content of specific elements and specify the type of requested answer elements. However, the complexity and the expressiveness of content-and-structure query languages are difficult for the end users because they have to know the logical organization of the document when expressing their information needs. Trotman and Lalmas (Trotman and Lalmas, 2006) showed that the structure did not improve the effectiveness of the retrieval system very much because users were normally not capable of giving useful structural hints with respect to INEX-IEEE collection. However, the content-and-structure query can be very useful for expert users in specialized scenarios.

3.1.3 The Narrowed Extended XPath I

The Narrowed Extended XPath I (NEXI) query language was developed at INEX (Trotman and Sigurbjörnsson, 2004a), as a simple query language for content-oriented XML retrieval evaluation. The enhancement comes from the introduction of a new function, named "about()". The "contains()" function of XPath, which requires an element (its text) to contain the given string content, was replaced by the "about()" function, which requires an element to be about the content. The NEXI query provides support for the descendant axis as follows: *//T[t]* is simple elements with paths matching T and contents about t. *//S[s]//T* returns elements T which are descendants of the element S, where the element S contains s. *//S[s]//T[t]* returns elements T which are descendants of the element S, where the element S contains s and the element T contains t. For example, the following information need of the query topic "2010010". The "title" indicating the keyword search contains only "director fearless jet li" and the "castitle" indicates the number of different categories that will be

processed **movie/title** “fearless” AND **movie/actor** “Jet Li” as shown in Figure 4.

```
<topic id="2010010" ct_no="13">
  <title>
  director fearless jet li
  </title>

  <castitle>
  //movie[about(.//title, fearless) AND about(.//actor, Jet Li)]//director
  </castitle>

  <description>
  Who is the director of the movie "Fearless" stared with "Jet Li"?
  </description>

  <narrative>
  I want to know the director of the movie "Fearless" stared with "Jet Li".
  </narrative>
</topic>
```

Figure 4 An example of NEXI topic 2010010

4. XML Ranking Methods

The Classical IR approach does not utilize any information about the XML document structure. Queries used by this approach are Content-only queries, which mainly contain a bag of words. The granularity of answers in this case is fixed; whole documents are usually returned to the user’s request. In contrast, XML information retrieval also determines the appropriate level of element granularity instead of offering whole documents as units of retrieval (Fuhr *et al.*, 2008; Kamps *et al.*, 2009; Geva *et al.*, 2010). The granularity of an element depends on the maximum coverage and density of the query terms within the XML document (Sigurbjörnsson and Kamps, 2005; Ogilvie and Callan, 2006; Mass and Mandelbrod, 2006).

The XML retrieval research community has been trying to solve the problem of

the appropriate element level in answering a query because the queried information lies in the smallest unit. For example, the user's query is "xml" in Figure 1. Both "title" of "article" and "section" elements are returned in the relevance list because they contain the matching term "xml" (Figure 5). Nevertheless, the "title" element is too small (it is the main theme of the element but the component is too small a unit of information or contains less information), and the parent nodes ("article" or "section") are given preference to the user.

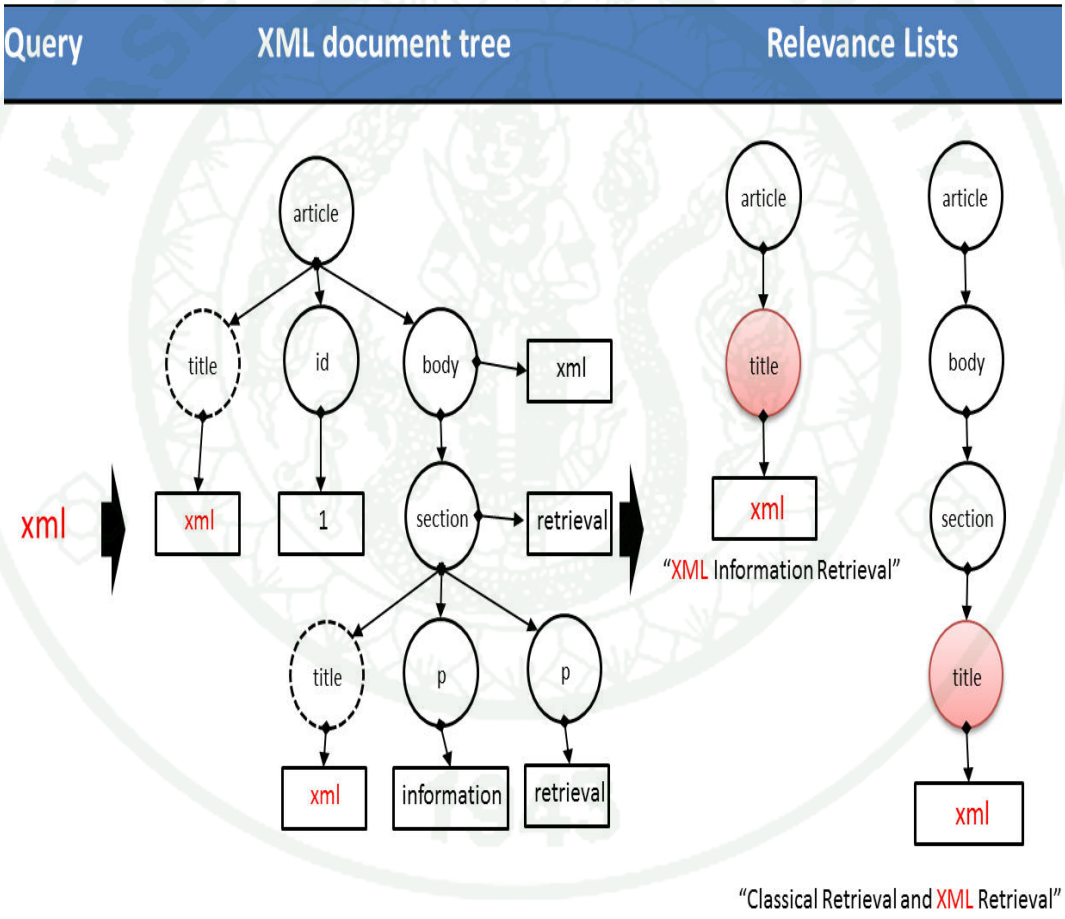


Figure 5 An illustration of a small unit of information

XML-IR always presents the most specific part (element) of a document, instead of the whole document. A more crucial problem of the XML retrieval system is its high redundancy since terms are repeated in each nested element (elements that are

contained within other elements) (Amer-Yahia and Lalmas, 2006; Manning *et al.*, 2008; Lalmas, 2009). For example, in indexing all XML elements when the user's query is "information", the system would return all the elements on the path between the root node and "information": "article", "body", "section" and "p", as shown in Figure 6. Returning redundant nested elements in a relevance list is not very user-friendly (Manning *et al.*, 2008).

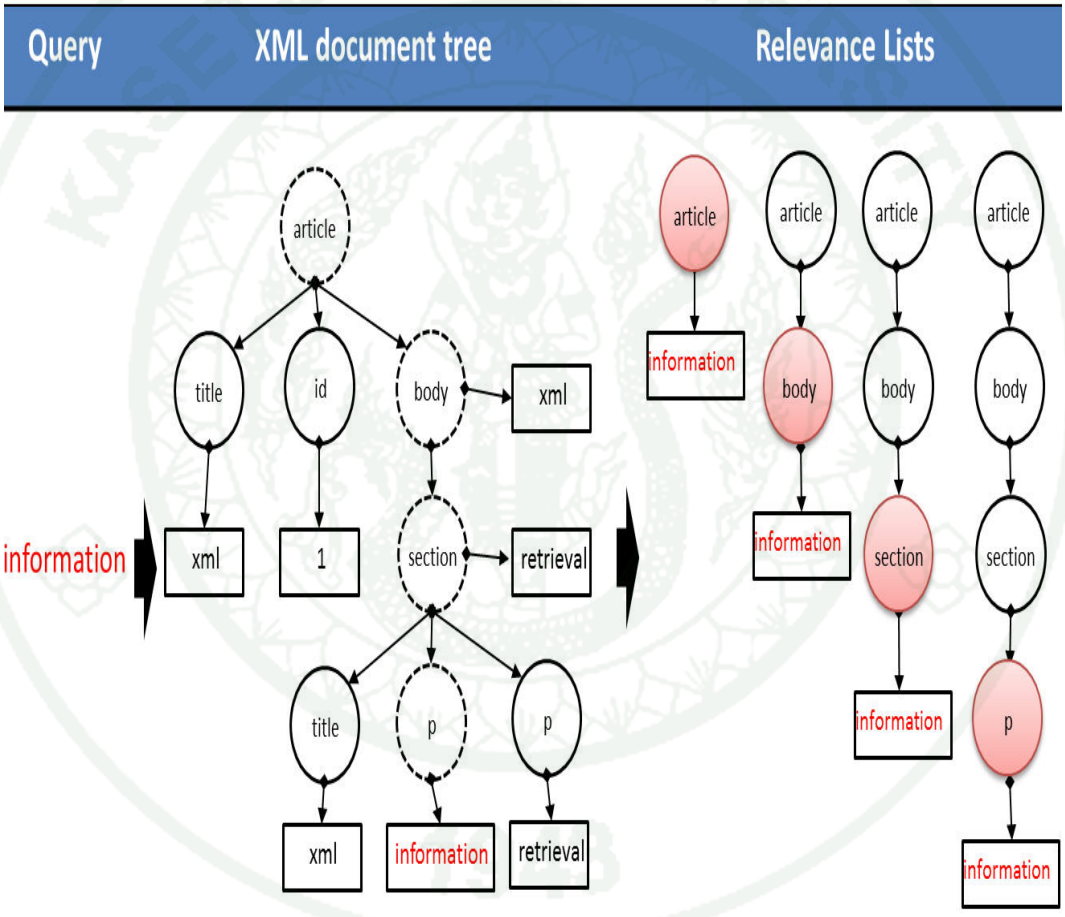


Figure 6 Illustration of redundant nested elements

In some cases, relevance sets still contain nested elements when all elements arer returned in response to a user query. Thus, XML retrieval systems remove some elements to reduce redundancy by collapsing several nested elements in the relevance list in a pre-processing stage (indexing time) and post-processing (querying time).

For example, if the user's query is "xml information", the "title" and "p" elements both turn up in the relevance list because they contain the matching terms "xml" and "information". In this situation, collapsing them to the "section" element would be beneficial to the user and this is therefore the preferred action, as shown in Figure 7.

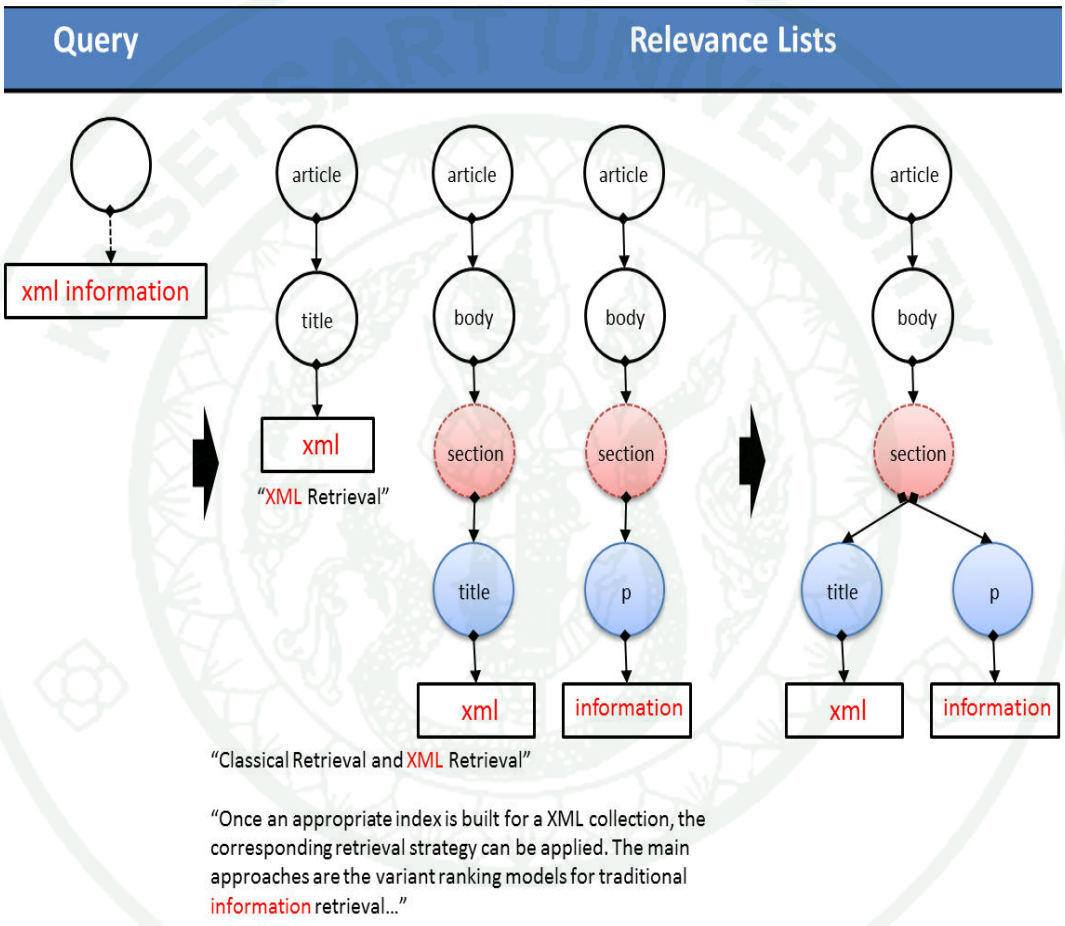


Figure 7 Illustration of collapsing units of information

Classical IR usually used TF and IDF to score relevant documents in response to a given query. Likewise, XML retrieval requires similar TF and IDF scoring, but at the element level. This raises an issue because of the nested elements of an XML document: the IDF value of a term will consider both the element that contains the term and its parents (Figure 8). It is possible to make straightforward modifications to standard IR methods specifically designed for XML retrieval. Each XML ranking

technique has its own advantages and disadvantages as shown in Table 1. We thus briefly describe XML retrieval approaches in the following section.

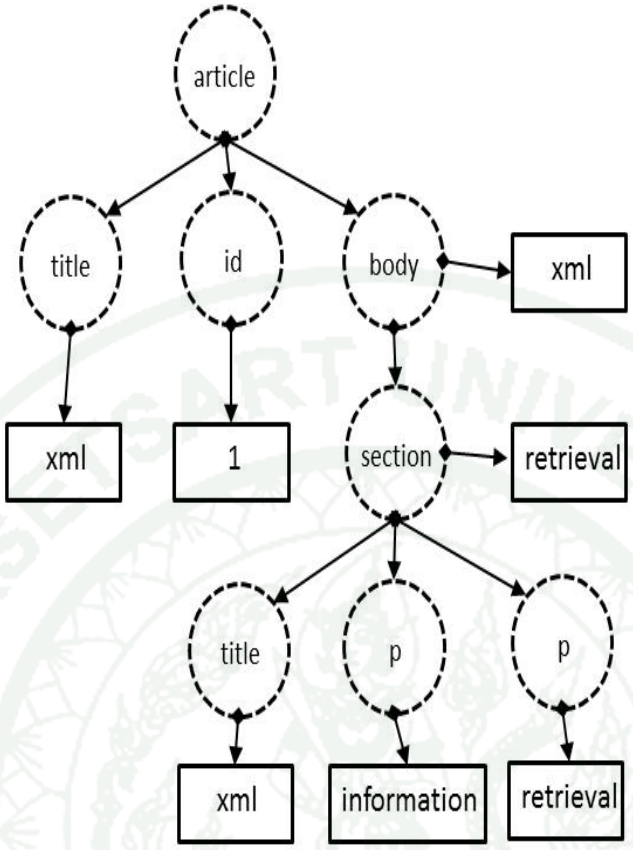
Table 1 Advantages and disadvantages of XML Ranking methods.

Ranking Method	Advantage	Disadvantage
Element base	-Easy to implement the system	-A major is highly redundant -Requires more storage space -Misses small elements -Raises nested elements issue
Combination base	-Overcomes the redundancy issue -Overcomes small elements issue -Overcomes storage space issue -Easy to implement the system	-Efficiency issue for the combined time -Raises nested elements issue
Selected type base	-Overcomes the redundancy issue -Overcomes small elements issue -Overcomes storage space issue -Overcomes nested elements	-Requires an expert -Requires training data and evaluation set for the tuning parameter

4.1 Element based approaches

These approaches allow each element to be indexed based on both the direct text and the text of descendants (by element based indexing (Sigurbjörnsson *et al.*, 2004)) and brute-force filtering of the output to remove nested elements, which is performed at query time. However, these strategies have a major drawback in that they result in high redundancy. Text occurring at the n-th level of the XML tree is indexed *n* times such that the text "information" is indexed in all the elements, viz. the "article", "body", "section" and "p" elements. This strategy, which is illustrated in Figure 8, requires more storage space.

One approach, called Element Scoring (Sigurbjörnsson and Kamps, 2005), aimed to return the element node without collapsing and also tried to solve the problem of nested elements by filtering them in the relevance list during query time. This strategy is based on a multinomial language model that combines the language models for the element and for the whole collection. The matching model of each element *e* calculates the probability that the element *e* and the collection *C* generates any of the query terms, and the elements are ranked according to the probability. This strategy employs element



Element ID	Element	Context
1	/article[1]	xml, 1, xml, information, retrieval
2	/article[1]/body	xml, information, retrieval
3	/article[1]/body[1]/section[1]	xml, information, retrieval
4	/article[1]/title[1]	xml
5	/article[1]/body[1]/section[1]/title[1]	xml
6	/article[1]/body[1]/section[1]/p[1]	information
7	/article[1]/body[1]/section[1]/p[2]	retrieval
8	/article[1]/@id[1]	1

Figure 8 An example of Element based indexing

based indexing. Given a query q, terms t_i for each element e and its corresponding element language model $\ominus_e$, the element e is ranked as follows:

$$P(q \mid \ominus_e) = \prod_{i \in q} \lambda P(t_i \mid e) + (1 - \lambda) P(t_i \mid C) \tag{7}$$

where

$P(t_i \mid e)$ is the probability of term t_i in element e.

$P(t_i \mid C)$ is the probability of term t_i in collection C.

λ is a smoothing parameter.

To account for the length of an element e, and in particular for the heavily biased distribution of small elements in XML documents, which can be used to set $P(e)$ as follows (Kamps *et al.*, 2004 2005a).

$$P(e) = \frac{length_e}{\sum_C length_e} \tag{8}$$

where

$length_e$ is the length of element e.

$\sum_C length_e$ is the length of element e occurring in collection C.

The assignment of a numeric score to an element e given a query q can be represented as follows:

$$Score(e,q) = P(e) * P(q \mid \ominus_e) \tag{9}$$

where

$Score(e,q)$ measures the relevance of element e to query q.

$P(e)$ is the probability of relevance for element e.

$P(q \mid \ominus_e)$ is the probability of the query q generated by language model $\ominus_e$.

Crouch proposed the Flexible Retrieval (Flex) approach, which produces a rank-ordered list of retrieved elements (Khanna, 2005; Crouch, 2006). This is equivalent to the result produced by the same retrieval against all element indices of the collection while filtering the output to remove nested elements. Flex is based on a modification of the basic vector space model with Pivoted Length Normalization (in the realm of elements that are not of the same length; longer elements are increased and smaller elements are decreased) (Singhal *et al.*, 1996) as follows.

$$Score(e,q) = \sum_{t \in q} W_{t,e} * W_{t,q} \tag{10}$$

$$W_{t,e} = \frac{\frac{1+\log[tf_{e,t}]}{1+\log[avg_{tf}]}}{(1-slope)+slope*(U/pivot)} \tag{11}$$

$$W_{t,q} = \frac{(1+\log[tf_{e,t}])*\log\left[\frac{N}{d_t}\right]}{(1-slope)+slope*(U/pivot)} \tag{12}$$

where

$Score(e,q)$ measures the relevance of element e to a query q.

$tf_{e,t}$ is the frequency of term t occurring in element e.

avg_{tf} is the average term frequency of all terms.

U is the number of unique terms.

$slope$ and $pivot$ are adjustments to the normalization factor (according to (Khanna, 2005) in INEX-2004 are $slope_{Paragraph}$ = 0.06, $pivot_{Paragraph}$ = 24, $slope_{Sub-section}$ = 0.01, $pivot_{Sub-section}$ = 92, $slope_{Section}$ = 0.10, $pivot_{Section}$ = 154, $slope_{Article}$ = 0.03, $pivot_{Article}$ = 538).

Nevertheless, these approaches are based on element based indexing. The disadvantage of the all-element index is the redundancy created due to nested elements

and also the large size of the files created.

4.2 Combination based approaches

These approaches address the problem of small elements by building the index from only the leaf elements. However, these strategies raise the issue of nested elements. In order to resolve the nested elements issue, these approaches are extended by using the combination of the weights of leaf elements with those of non-leaf elements (collapsing several nested elements), which is performed at query time. This strategy is illustrated in Figure 9, where the leaf elements are indexed. These approaches avoid returning nested elements in the result list in the ways described below.

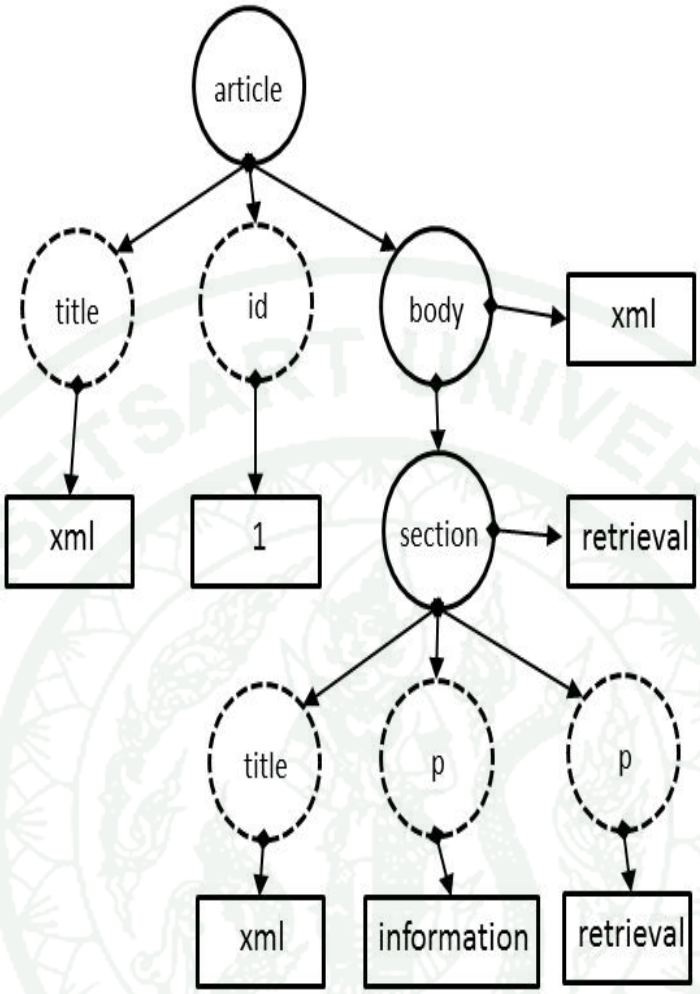
Ogilvie and Callan proposed the Score Aggregation technique, which ranks elements based on the aggregated representation of the term statistics of its own context with the statistics of the element's children (Ogilvie and Callan, 2006). The effectiveness of the aggregation depends on β parameters that set the weight for each component and are estimated through learning methods. This approach tries to solve the nested element issue by using the aggregation of the weights of leaf elements and those of non-leaf elements, which is performed at query time. Afterwards, the weights of non-leaf element are compared to those of leaf elements. If their weights are higher than those of their children, then all the leaf elements are removed from the relevant list. The assignment of a numeric score to a document for a query can be represented as follows:

$$Score(e,q) = \beta_e * P(q \mid \ominus_e) + \sum_{c \in e} \beta_c * P(q \mid \ominus_c) \tag{13}$$

$$\beta_e + \sum_{c \in e} \beta_c = 1 \tag{14}$$

where

$Score(e,q)$ measures the relevance of element e to a query q.



Element ID	Element	Context
1	/article[1]/title[1]	xml
2	/article[1]/body[1]/section[1]/title[1]	xml
3	/article[1]/body[1]/section[1]/p[1]	information
4	/article[1]/body[1]/section[1]/p[2]	retrieval
5	/article[1]/@id[1]	1

Figure 9 An example of Leaf-Only indexing

c is the child c of element e.

β_e is the β for the element e.

β_c is the β for the child c of element e.

This model for the β parameters includes the contribution of each element in the aggregation. The drawback of this model involves the estimation of these β values, which are usually estimated by learning methods (Ogilvie and Callan, 2006).

Geva (Geva, 2007) presented the Garden Point XML Retrieval (GPX) technique, which ranks elements based on leaf-node indexing and assigns weight by propagating upwards to ancestors (Geva, 2005). The resulting relevance score for each element is a weighted accumulation of ranking scores of an element's children. The GPX strategy is a propagation method that was proposed as a Bottom-Up Scheme (BUS) (Dongwook *et al.*, 1998) to handle nested elements. This description is based on the version used for INEX 2004 (Geva, 2005), INEX 2005 Ad hoc track in the INEX-IEEE collection (Geva, 2006) and for INEX 2006 on the Ad hoc track in the INEX-Wikipedia collection. A leaf elements relevance score is calculated as follows:

$$Score(e_c, q) = N^{n-1} * \sum_{t \in q} \frac{tf_{e,t}}{Cf_t} \tag{15}$$

where

$Score(e_c, q)$ measures the relevance of child c of element e to a query q.

n is the number of unique query terms contained within the element e.

N is the smoothing parameter (N = 5 according to (Geva, 2005)).

$tf_{e,t}$ is the frequency of term t occurring in element e.

Cf_t is the frequency of term t occurring in collection C.

The assignment of a numeric score to the non-leaf elements e given a query q can be represented as follows:

$$Score(e,q) = D(m) * \sum_{c \in e} score(e_c, q) \tag{16}$$

where

$Score(e,q)$ measures the relevance of element e to a query q.

c is the child c of element e.

$D(m)$ is the smoothing parameter set as follows (according to (Geva, 2007)).

$$D(m) = \begin{cases} 0.49 & \text{; if } e \text{ has one child } c \\ 0.99 & \text{; otherwise} \end{cases} \tag{17}$$

Because of the smoothing parameter $D(m)$, When an element has only one relevant child element, the child is ranked higher than its parent; otherwise, the parent element ranks higher than the child.

Arvola et al. (Arvola *et al.*, 2005 2011) proposed the General contextualization function in their work on the Tampere information retrieval and indexing of XML (TRIX). This strategy solved the nested elements issue based on a general re-weighting method in which any context of the element along the hierarchical path may influence the weight of the element as follows:

$$Score(t,q) = \frac{tf_t}{v * \left((1-b) + b * \frac{ef_c}{ef_t}\right)} * \frac{\log[N - d_t]}{\log[N]} \tag{18}$$

$$Score(e,q) = \sum_{i=1}^{n} \frac{Score(t_i,q)}{n} \tag{19}$$

In terms of the contextualization function, the context levels can be taken into account in three ways, which are Root, Parent and Tower (the latter being the element between Root and Parent nodes) as follows:

$$Root(e,q) = \frac{Score(e,q) + 1.5 * Score(\partial_1(e),q)}{2.5} \tag{20}$$

$$Parent(e,q) = \frac{Score(e,q) + Score(\partial_{len(e)-1}(e),q)}{2.0} \tag{21}$$

$$Tower(e,q) = \frac{\sum_{i=1}^{len(e)} Score(\partial_i(e),q)}{len(e)} \tag{22}$$

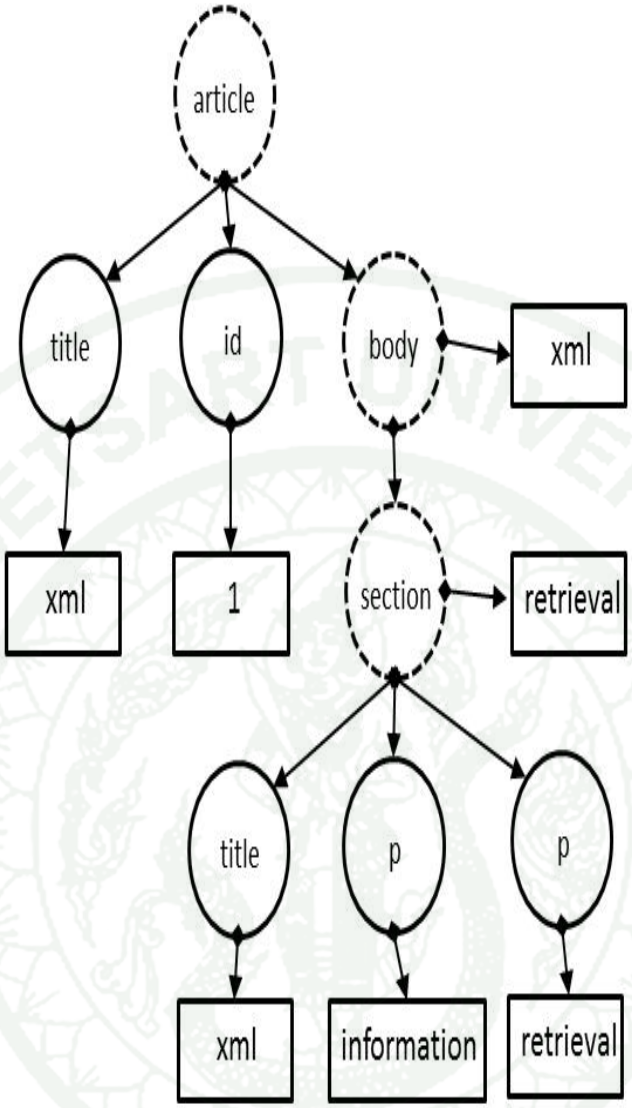
where

$Score(e,q)$ measures the relevance of element e to a query q.
$Root(e,q)$ measures the relevance of Root of element e to a query q.
$Parent(e,q)$ measures the relevance of Parent of element e to a query q.
$Tower(e,q)$ measures the relevance of Tower of element e to a query q.
ef_c is the number of all the descendant content elements e.
ef_t is the number of all the descendant content elements e containing term t.
v and b are constants for tuning the weighting.
∂ is the cutting operation with the length of element e.

The result of their experiment showed that the root contextualization $Root(e,q)$ worked best. However, the root might carry large size elements and those elements may not be relevant to the user (Pharo, 2008).

4.3 Selected type based approaches

These approaches try to solve the problem of small units of information and nested elements by selected types using a threshold element size and semantic elements. This strategy is illustrated in Figure 10, where the elements are indexed. The relevance score is estimated from the statistics of a term at the selected element level, and elements in the result lists are not collapsed. This approach is based on the restriction that only element types that a system designates as useful search results will



Element ID	Element	Context
1	/article[1]	xml, 1, xml, information, retrieval
2	/article[1]/body	xml, information, retrieval
3	/article[1]/body[1]/section[1]	xml, information, retrieval

Figure 10 An example of Selective or Distributed indexing

be indexed (Manning *et al.*, 2008). However, this strategy raises the issue of manually choosing all potentially retrievable elements.

4.3.1 Extended vector space model

The extended vector space model, called Score Merging, presented in the JuruXML (Mass and Mandelbrod, 2006), builds an index for different types of elements with the aim of retrieving small elements but avoiding return of nested elements. The ranking model must run each index separately and it retrieves ranked lists of elements. The scores from the different indices are not directly comparable because they are based on separate vector spaces with different numbers of elements and different average element lengths. Accordingly, the scores are normalized by dividing them by the maximum score of the query. These lists are merged to provide a single rank across all element types. Lists are merged so that scores across indices are comparable. Indexing is separately created for each type of element as shown in Figure 10. The assignment of a numeric score to a document for a query can be represented as follows:

$$score(e,q) = \frac{\sum_{e \in c} W_{t,q} * W_{t,e} * ief_t}{length_e * length_q} \tag{23}$$

$$W_{t,q} = \log\left[\frac{tf_{t,q}}{avglen_q}\right] \tag{24}$$

$$W_{t,e} = \log\left[\frac{tf_{t,e}}{avglen_e}\right] \tag{25}$$

$$ief_t = \log\left[\frac{N}{e_t}\right] \tag{26}$$

where

$Score(e,q)$ measures the relevance of element e to a query q.

$tf_{t,q}$ is the frequency of a term t occurring in a query q.
$tf_{t,e}$ is the frequency of a term t occurring in an element e.
N is the total number of an element in the collection.
e_t is the total number of elements in which the term t occur.
$avglen_q$ is the average number of occurrence of all query terms.
$avglen_e$ is the average length of an element e in the entire collection.
$length_q$ is the number of unique terms in a query q.
$length_e$ is the number of unique terms in an element e.
ief_t is the inverse element frequency weight of term t.

4.3.2 Probabilistic and Language models

Broschart and Schenkel (Broschart and Schenkel, 2008) presented the proximity scoring function. This led to significant effectiveness improvements for XML retrieval called the Proximity-aware function. This strategy handles small elements by selected types using a threshold element size. This method introduces modified proximity scores as follows:

$$Score(e,q) = W_{q,e} + Prox_{q,e} \tag{27}$$

$$W_{q,e} = \sum_{t \in q} \frac{(k_1+1) * tf_t}{K + tf_t} * ief_t \tag{28}$$

$$ief_t = \log \left[\frac{N - e_t + 0.5}{e_t + 1} \right] \tag{29}$$

To compute the proximity part of the score for each term t, an accumulated interim score $acc_{t,i}$ for each query term t_i that depends on the distance of this term's occurrences in the element to other terms, adjacent query term t_i, for each adjacent occurrence of a term t_j at distance d to an occurrence of t_i, the $acc_{t,i}$ grows by

$\frac{ief_t}{d}$. The proximity part of an element's score is computed as follows.

$$Prox_{q,e} = \sum_{t \in q} \min\{1, ief_t\} \frac{(k_1+1) * acc_t}{K + acc_t} \tag{30}$$

where

$Score(e,q)$ measures the relevance of element e to a query q.

$Prox_{q,e}$ measures the proximity of element e to a query q.

acc_t is calculate by $\frac{ief_t}{d}$.

d is the distance of query terms occurrence in the element e.

Theobald et al. (Theobald and Weikum, 2002; Theobald *et al.*, 2005) presented the extended BM25 function in a $top-k$ retrieval engine for text and XML data called TopX (Theobald, 2006), which is optimized for the efficient retrieval of the $top-k$ results of the baseline BM25 (Stephen *et al.*, 1994) as follows:

$$Score(e,q) = \sum_{t \in q \cup e} \frac{(k_1+1) * tf_{t,e}}{k_1 * \left((1-b) + b * \frac{len(e_A)}{avel_A}\right) + tf_{t,e}} * ief_t \tag{31}$$

$$ief_t = \log \left[\frac{\left(\frac{N_A - e_t + 0.5}{e_t}\right)}{N_A + 0.5} \right] \tag{32}$$

where

$len(e_A)$ is the length of element e with tag A.

$avel_A$ is the average length of elements in the entire collection with tag A.

k_1 and b is a common tuning parameter for the BM25.

ief_t is the inverse element frequency weight of term t.

N_A is the total number of an element A in the entire collection.

The modified function provides the influence of the $tf_{t,e}$ with tag A (Theobald *et al.*, 2008). However, this strategy is limited in that each tag name must be the same name to implement automatic grouping and weight calculation.

Robertson et al. (Stephen *et al.*, 2004) presented the ranking function BM25F, which is composed of several document fields with potentially different degrees of importance; these fields are known as selected fields, and they give substantial improvements over the baseline BM25 (Stephen *et al.*, 1994). This strategy also involves eliminating small elements by using semantic elements, an operation which is performed manually. Suppose we have fields $\overbrace{f_1, f_2, \cdots, f_n}^{n}$ in a given document d, and term t has frequency $tf_{d,t,f}$ in field f. Being a linear weighted combination of term frequencies for each field f, the BM25F function (Stephen *et al.*, 2004) is calculated as follows:

$$BM25F(d,q) = \sum_{t \in q} \frac{(k_1+1) * tf_{d,t,f}}{K + tf_{d,t,f}} * idf_t \tag{33}$$

With field weights W_f, these are modified as follows:

$$tf_{d,t,f} = \sum_{f} W_f * tf_{d,t,f} \tag{34}$$

where

idf_t is the inverse document frequency weight of term t.

Lu et al. (Lu *et al.*, 2005) proposed that the BM25 be extended by adding the weight for each element. This modification, called the BM25E function, applies field weighting in a straightforward manner to the XML element as follows:

$$BM25E(e,q) = \sum_{t \in q} \frac{(k_1+1) * tf_{d,t,e}}{K + tf_{d,t,e}} * idf_t \tag{35}$$

With field weights W_f, these are modified as follows:

$$tf_{d,t,e} = \sum_{f \in e} W_f * tf_{d,t,e} \tag{36}$$

Table 2 The top five participants in the Ad Hoc Track Focused Task of INEX 2009.

No.	Authors	Ranking Method	iP[0.01]
1	(Itakura and Clarke, 2010b)	BM25F	0.6333
2	(Buffoni *et al.*, 2010)	BM25	0.6141
3	(Broschart and Schenkel, 2010)	BM25 & Proximity	0.6134
4	(Géry and Largeron, 2010)	BM25	0.6060
5	(Koolen *et al.*, 2009)	Element Scoring	0.5997

The authors constructed three fields manually: “title”,“abs”, “st” elements in the INEX-IEEE collection, and showed that the tuning values for W_f were all integers, and they tuned W_f $(title, abs, st)$ from $(1,1,1)$ to (max, max, max), where max is the limit of increasing value in the experiment using increments of 1. The result showed that the values W_{title} = 2356, W_{abs} = 4, and W_{st} = 22 returned the highest average precision score. Using the BM25F scoring function presented in (Itakura and Clarke, 2010a; Pérez-Agüera *et al.*, 2010). Itakura and Clarke constructed two fields manually, one for “title” and another for “body” (Itakura and Clarke, 2010a). The “title” field consists of the concatenation of an article title and any ancestral and current section titles. The “body” field contains the rest of the text in the element. Thereafter, the values for W_f $(title, body)$ are tuned from $(1,1)$ to (max, max) using increments of 0.1. The result showed that the values of $(4.0, 1.2)$ for W_f achieve the highest result on iP[0.01]. Along these lines, another study (Perez-Aguera:2010:UBS) constructed five fields manually; “text”,“title”, “inlinks”,“obj”, “type” on the Resource Description Framework (RDF) document structure and the values of tuning W_f are W_{text} = 1, W_{title} = 3, $W_{inlinks}$ = 2, W_{obj} = 2, W_{type} = 2. The BM25F function has performed well in past evaluations as shown in Table 2.

Many researchers make use of the BM25F function (Stephen *et al.*, 2004; Craswell *et al.*, 2005; Lu *et al.*, 2005; Najork *et al.*, 2007; Stephen and Hugo, 2009; Itakura and Clarke, 2010a; Pérez-Agüera *et al.*, 2010). Unfortunately, all

elements would contribute to a given element e. The fact that, in practice, there are more than ten-million elements in each INEX collection makes it very difficult to tune W_f for each element e. As a result, there are issues that require additional attention. Firstly, which XML elements should be treated as fields? Secondly, what appropriate weight should be assigned to each field? Previously, document fields were selected manually, and the weight for each chosen field was tuned before being assigned.

5. XML Evaluation Methods

Evaluation the effectiveness of ranking techniques is very important in information retrieval. The two main objectives of an information retrieval system are coverage and accuracy.

5.1 Normalized extended Cumulated Gain (nxCG)

The XML-specific metrics were newly introduced for the INEX-IEEE benchmark: the normalized extended Cumulated Gain (nxCG) metrics are an extension of the Cumulated Gain (CG) metrics (Järvelin and Kekäläinen, 2002) that consider the dependency of XML elements within an evaluation. Kazai and Lalmas defined relevance along two dimensions:-exhaustivity and specificity (Kazai and Lalmas, 2006). An element is exhaustive to a query if it contains all the required information; an element is specific to a query if all its content concerns the query.

A multiple degree relevance scale was necessary to allow the explicit representation of how exhaustively a query is discussed within an element with respect to its child elements. INEX adopted a four-level relevance scale as follows: *Irrelevant (0)*, the element does not contain any information about the query; *Marginally relevant (1)*, the element mentions the query, but only in passing; *Fairly relevant (2)*, the element discusses the query, but not exhaustively; *Highly relevant (3)*, the element discusses the query exhaustively. A four-level relevance scale for component coverage was therefore also adopted: *No coverage (N)* is not a theme of the element; *Too large (L)* is only a minor theme of the element; *Too small (S)* is the main theme of the element but the component is too small a unit of information; *Exact coverage (E)* is the main theme of

the element as a meaningful unit of information.

(a)

ID	XPath	$quant_e$
1	/article[1]/body[1]	0.25
2	/article[1]/body[1]/section[1]	1.50
3	/article[1]/title[1]	1.25
4	/article[1]/body[1]/section[1]/title[1]	1.00
5	/article[1]/body[1]/section[1]/p[1]	0.75

(b)

ID	$quant_e$	xIG
2	1.50	1.50
3	1.25	2.75
4	1.00	3.75
5	0.75	4.50
1	0.25	4.75

(c)

ID	$quant_e$	xCG	nxCG
5	0.75	0.75	0.50
3	1.25	2.00	0.75
-	0.00	2.00	0.53
-	0.00	2.00	0.44
1	0.25	2.25	0.47

(d)

r	MAnxCG
1	0.500
2	0.610
3	0.580
4	0.547
5	0.532

Figure 11 Illustrations of the *xIG*, *xCG*, *nxCG* and *MAnxCG* calculation

Each element result gets a score from the assessment values, so a result list is mapped to a list of scores, the quantization $quant_e$ function. The quantization function of an assessed element e, given by the combined values of exhaustiveness (E) and specificity (S), which are then denoted by pairs $e,s \in ES$, where $ES = \{(0,N),(1,L),(1,S),(1,E),(2,L),(2,S),(2,E),(3,L),(3,S),(3,E)\}$, and referred to as $quant_e$ which are defined as follows:

$$quant_e = \begin{cases} 1.00; & \text{if } (3,E) \\ 0.75; & \text{if } (2,E),(3,S),(3,L) \\ 0.50; & \text{if } (1,E),(2,S),(2,L) \\ 0.25; & \text{if } (1,L),(1,S) \\ 0.00; & \text{if } (0,N) \end{cases} \tag{37}$$

Thus, the cumulated gain reached by a retrieval engine is divided by the corresponding cumulated gain in the ideal recall vector (that is, the vector obtained by ordering the assessed elements by decreasing relevance). Given $xCG(i)$ as the cumulated gain at rank i and $xIG(i)$ as the cumulated gain of the ideal run, $nxCG(i)$ is defined as follows (Kazai and Lalmas, 2006):

$$xCG(i) = \sum_{j=1}^{i} quant_{e,j} \tag{38}$$

$$xIG(i) = \sum_{j=1}^{i} quant_{e,j} \tag{39}$$

$$nxCG(i) = \frac{xCG(i)}{xIG(i)} \tag{40}$$

For each point, the gain values for higher ranks should have a higher weight. *Mean Average nxCG* at rank r addresses this by calculating the average $nxCG$ values for ranks $\overbrace{1,2,\cdots,r}^{r}$:

$$MAnxCG(r) = \frac{1}{r} * \sum_{i=1}^{r} nxCG(i) \tag{41}$$

To illustrate, assume $quant_e$ for each relevance element as shown in Figure

11(a). We give an example of *xIG* in Figure 11(b), Figure 11(c) shows the *xCG* and *nxCG*. Afterwards, the *MAnxCG* calculation is as shown in Figure 11(d).

5.2 Mean Average interpolate Precision (MAiP)

ID	XPath	Size
1	/article[1]/body[1]	300
2	/article[1]/body[1]/section[1]	150
3	/article[1]/title[1]	100
4	/article[1]/body[1]/section[1]/title[1]	100
5	/article[1]/body[1]/section[1]/p[1]	50

ID	Size	Precision	Recall (r = 700)
5	50	1.00	0.07
3	150	1.00	0.28
-	200	0.50	0.28
-	175	0.34	0.28
-	150	0.27	0.28

Figure 12 Illustrations of P_r and R_r calculation

INEX was determined to evaluate effectiveness based on the amount of relevant information retrieved measured in terms of the length of relevant elements instead of counting the number of relevant elements retrieved (Kamps *et al.*, 2007b). The evaluation framework is based on the amount of relevant text in elements (Pehcevski and Thom, 2005 2006). A measurement of selected precision at rank *r*

is defined as follows:

$$P_r = \frac{\sum_{i=1}^{r} rsize(p_i)}{\sum_{i=1}^{r} size(p_i)} \tag{42}$$

To achieve a high precision score at rank r, the document parts are retrieved. Recall at rank r is defined as follows:

$$R_r = \frac{\sum_{i=1}^{r} rsize(p_i)}{Trel_q} \tag{43}$$

where

p_r is the document part assigned to rank r returned to a query q.
$rsize(p_r)$ is the length of relevant text contained by p_r.
$size(p_r)$ is the length of text contained by p_r.
$Trel_q$ is the total amount of relevant text for a query q.

An example of P_r and R_r calculation is shown in Figure 12. To achieve a high recall score at rank r, the document parts retrieved need to contain as much relevant text as possible. An issue with the precision measured by P_r given in function: (42), an iP_x is calculated by interpolating precision scores at recall levels:

$$iP_x = \begin{cases} max_{1 \le r \le |L_q|} P_r \wedge R_r \ge x & ; x \le R[|L_q|] \\ 0 & ; x > R[|L_q|] \end{cases} \tag{44}$$

where

$R[|L_q|]$ is the recall over all documents retrieved. For example, $iP[0.01]$ calculates interpolated precision at the 1% recall level for a given topic.

Overall performance measure scores are based on the measure of Average interpolated Precision (AiP). For each query, the AiP is calculate by averaging the interpolated precision scores calculated at ranks $\overbrace{1,2,\cdots,101}^{101}$ standard recall levels:

$$AiP = \frac{1}{101} * \sum_{0.00}^{1.00} iP_x \tag{45}$$

Performance across a set of queries is measured by calculating the mean of the AiP values obtained by the measure for each individual query, resulting in Mean Average interpolate Precision (MAiP). Assuming there are $\overbrace{1,2,\cdots,n}^{n}$ queries:

$$MAiP = \frac{1}{n} * \sum_{t=1}^{n} AiP_t \tag{46}$$

6. XML Compression Methods

Recent developments in XML compression techniques are aimed at not only reducing XML data storage, but also increasing efficiency (Sakr, 2009). XML data compressors can be divided into two groups on the basis of to their ability to support queries: 1) Non-Queriable XML Compressors, which have the goal of achieving the highest compression ratio, and 2) Queriable XML Compressors, which permit performing direct queries on compressed data. Several compression strategies developed in XML are as described below:

6.1 Non-Queriable XML Compressors

Non-Queriable XML compressors aim to achieve the highest compression ratio.

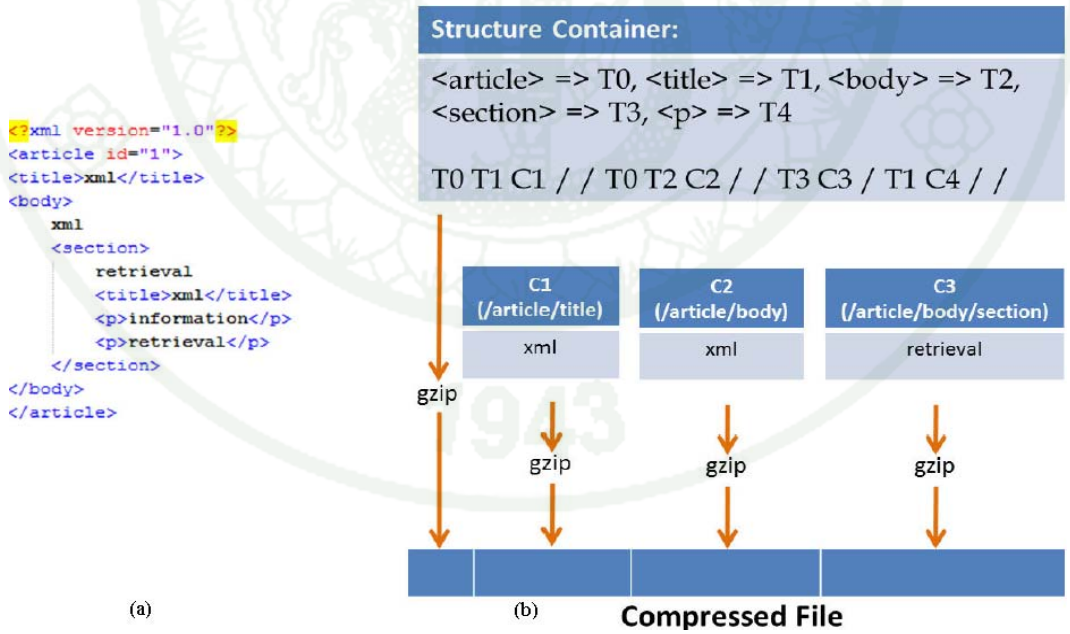


Figure 13 An illustrations of XMill compression processing

6.1.1 XMill Compressor

XMill (Liefke and Suciu, 2000), a technique which compresses both data and tags for size reduction, starts by separating the tag (which is composed of elements and attributes) from the data (which are characters). The relationships of the data groups are then organized with similar data placed in the same group. Following that, data compression is effected using gzip (Gailly and Adler, 2010). For the data to appear in the same file, grouping requires understanding of the data definitions, which, in turn, are dependent on the application type. XMill allows the user to check the data definition. The disadvantage of XMill is that the compressed data does not facilitate a data search. Nevertheless, the XMill compressor can be considered the first research work that drew the attention of researchers to the importance of data storage, its inherent problems and how such problems might be solved through XML data compression. Data that have been compressed are not in the form of the XML schema structure. Unfortunately, this technique has a parse error with the symbol "$>$" expected after "/" in the element. For example, an XML document is shown in Figure 13(a), and the XML compressed data in Figure 13(b).

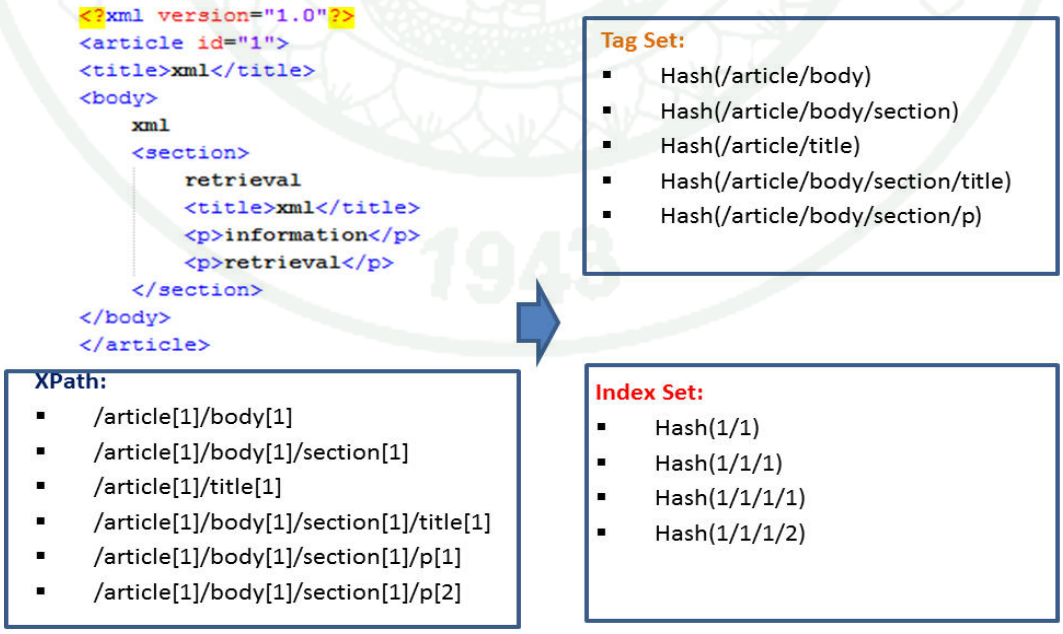


Figure 14 An illustration of GPX compression processing

6.1.2 Garden Point XML Retrieval Compressor

The Garden Point XML Retrieval (GPX) (Geva, 2006) search engine uses a relational database and is implemented with an inverted list data structure. This can be represented by two expressions, namely a Tag-set and an Index-set, as shown in Figure 14. The original XPath can be reconstructed from the tag-set and the index-set. The GPX assigns a hash code to each tag set and each index-set, and creates auxiliary database tables mapping the hash codes to the corresponding tag-set and index-set entries.

6.1.3 XPACK Compressor

XPACK (Mairieng and Pluempitiwiriyawej, 2003) compresses XML data using grammatical approaches in data compression and decompression. The main component of XPACK is the Grammar Generator, which creates the grammar. The second component is the Compressor which compresses the data. The final component is the Decompressor, which decompresses the compressed data using the old structure of the data. However, XPACK cannot manage XML data that has mixed content elements (i.e. elements composed of elements and characters), thereby limiting users in searching for data in compressed XML. The Grammar of this technique defined as follows:

$$< t >= t\sharp / d(a, v) \tag{47}$$

where

t is the Open tags.

$\sharp$ is the content in tag t.

$/$ is the Open tags.

d is the data type.

a is an attribute in tag t.

v is the value of attribute a.

6.2 Queriable XML Compressors

Queriable XML compressors allow direct queries on compressed data. There are several variants as shown below.

6.2.1 XGrind Compressor

XGrind (Tolani and Haritsa, 2002) is a technique that compresses data and tags but permits the user to search for data after compression. This attribute results from the fact that the compressed data still maintains the structure of XML data. However, XGrind will compress only XML data that has DTD structure. Hence, for data without DTD structure, users have first to create DTD for the XML data set they wish to compress. As an example, we show an XML document in Figure 15(a), and the compressed XML data in Figure 15(b).

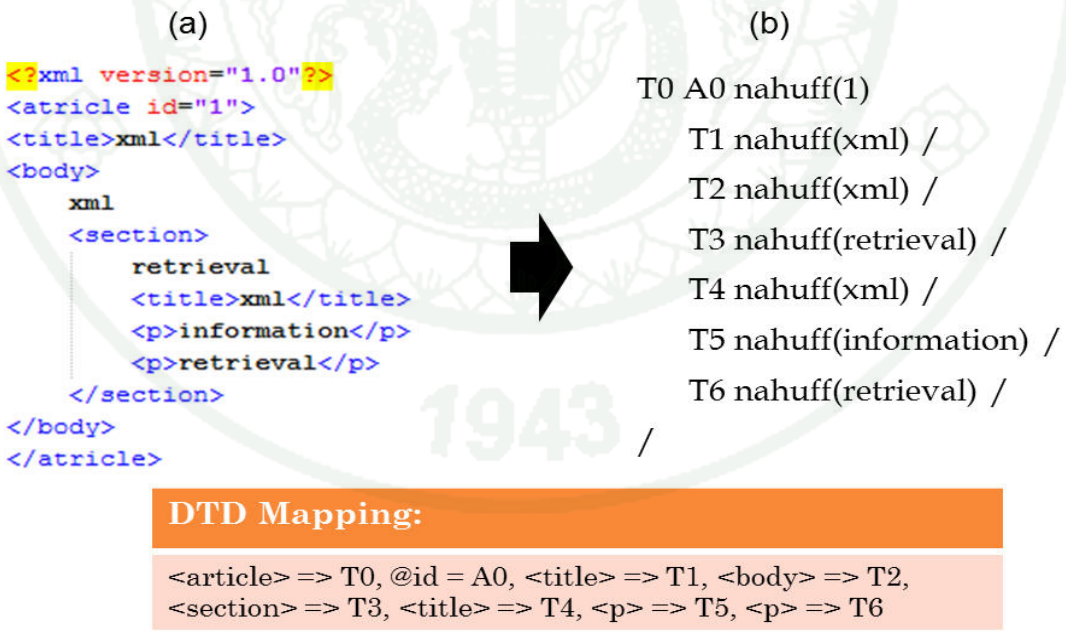


Figure 15 An illustration of XGrind compression processing

6.2.2 XSchemaTag Compressor

XSchematag (Wichaiwong and Jaruskulchai, 2007) is a technique that compresses only the XML tag while still permitting the search and maintenance of documents because the data are already in the form of XML. This research focused on the compression of tags in an electronic commerce application. Most data sent through web documents have many complicated elements. The main components of this technique are as follows:

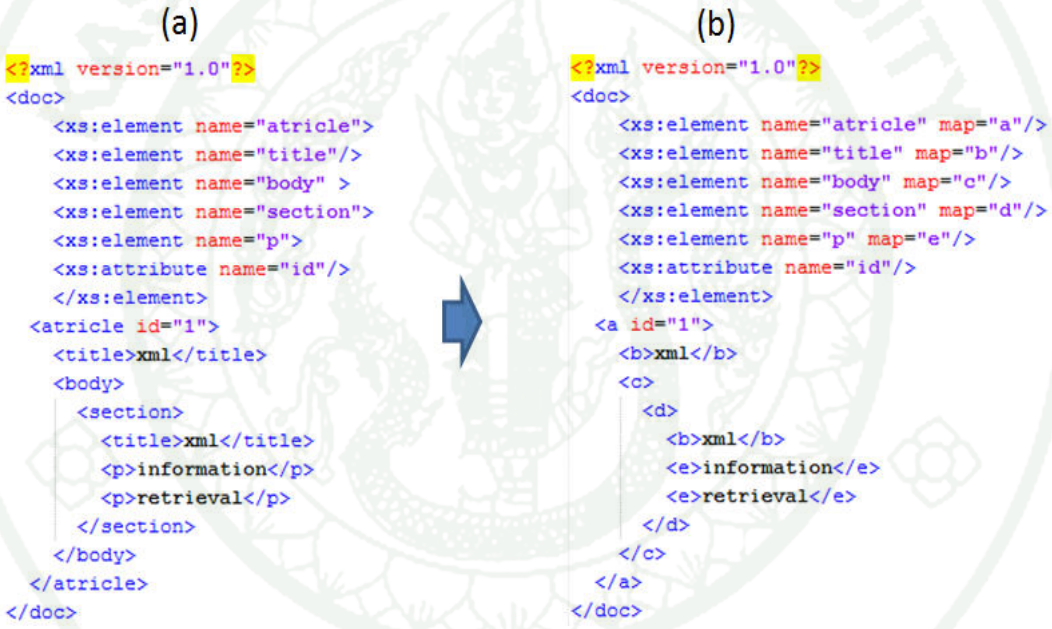


Figure 16 An illustration of XSchemaTag processing

The Schema tag Scanner reads the XML data structure by analyzing all the structures of all the XML data. The Compressor compresses the structure of XML Schemas. The compressed data can be used on the web in the future by using a dictionary based technique, and the Decompressor reads the structure of the XML schema and decompresses it back to the same data that had been received from the web by lossless compression. The advantage here is that the compressed data retains the old data structure. However, the XSchemaTag technique does not take into account the frequency of tags or the position of tags (or elements). As an example, an XML

document is shown in Figure 16(a), and the compressed of the XML data in Figure 16(b).

6.2.3 XPRESS Compressor

XPRESS (Min *et al.*, 2003) compresses both the data and the tag. Its advantage, like that of XGrind, is that it can search for the data after the compression. However, XPRESS does not use DTD. In addition, XPRESS adopts a novel approach that uses reverse arithmetic encoding, which is a method of organizing data to allow effective search for XPath expressions. Furthermore, XPRESS can determine the data type without the information being provided by the user. Nevertheless, the use of XPRESS is limited because it is not compatible with documents that use ID and IDREF. It is also not possible to decompress data back into normal XML.

6.3 Entropy and Redundancy

When data are compressed, the goal is to reduce the redundancy, and leave only the informational content. To measure the quantity of information within the data, Shannon defined the concept of entropy as follows (Shannon, 1948):

Let $H = \overbrace{x_1, x_2, \cdots, x_n}^{n}$ be the set of source units. Let p_i be the probability of occurrence of source unit $x_i \leq 1 \leq n$. The entropy of unit x_i is equal to

$$H_i = \log p_i \tag{48}$$

This definition has an intuitive interpretation: If $p_i = 1$, it is clear that x_i contains no information. Similarly, the smaller the value of p_i, the more unlikely x_i is to appear, and therefore its informational content is larger. The average entropy of a source unit from H is defined as follows:

$$AH = -\sum_{i=1}^{n} p_i \log p_i \tag{49}$$

The entropy of a source message $X = \overbrace{x_{i1}, x_{i2}, \cdots, x_{ij}}^{ij}$ from H as follows.

$$H(X) = -\sum_{i=1}^{n} p_i \sum_{j=1}^{m} p_{ij} \log p_{ij} \tag{50}$$

MATERIALS AND METHODS

Materials

1. Computer

The algorithms were implemented by using the *C♯* programming language. The experiments were tested under the following computer specification:

1. Intel Core i5 processor 750 * 2.67 GHz.
2. RAM 6 GB
3. Hard Disk 1,024 GB
4. OS Windows 7 Operating System
5. MySQL Community Server 5.5.23
6. Sphinx Search Server 2.0.4

2. XML Collections

1. The document collections were from the INEX-IEEE document collection (Geva *et al.*, 2011), which contained a total of 16,819 articles from 24 IEEE Computer Society journals covering the period of 1995-2005 and totaling 764 megabytes in size and 11 million elements in its canonical form.

2. The INEX-Wiki06 Corpus for English Wikipedia from early 2006 (Denoyer and Gallinari, 2006) contained 659,338 Wikipedia articles; the total size is 4.6 GB without images and 52 million elements. On average, an article contained 161.35 XML nodes, whereas the average depth of a node in the XML tree of a document was 6.72.

3. The INEX-Wiki09 (Schenkel *et al.*, 2007) collection was created from the October 8, 2008 dump of the English Wikipedia articles and incorporated semantic annotations from the 2008-w40-2 version of YAGO. It contained 2,666,190 Wikipedia

articles and had a total uncompressed size of 50.7 GB. There were 101,917,424 XML elements of at least 50 characters.

4. The IMDB data collection used in INEX 2010 (Geva *et al.*, 2011) was generated from the plain text files published on the IMDB web site on April 10, 2010. There were two kinds of objects in the collection, movies and persons involved in movies, e.g. actors/actresses, directors, producers and so on. Each object was richly structured. For example, each movie had the title, rating, directors, actors, plot, keywords, genres, release dates, trivia, etc.; and each person had his/her name, birth date, biography, filmography, etc. Each XML document contained information about one object, i.e. a movie or person (Trotman and Wang, 2011). In total, the IMDB data collection contained 4,418,081 XML documents, including 1,594,513 movies, 1,872,471 actors, 129,137 directors who did not act in any movie, 178,117 producers who did not direct nor act in any movie, and 643,843 other people involved in movies neither produced nor directed nor acted in any movie.

5. The INEX-WS-2010, a collection of Web Services Description Language (WSDL) documents (Geva *et al.*, 2011) were taken directly from real-world public Web services (Mani and Nagarajan, 2002; Booth *et al.*, 2004; Booth and Liu, 2007) and indexed by the Google search engine. The test collection was pre-processed so that only valid WSDL 1.1-compliant descriptions were retained for XML-based retrieval that contains 1,987 articles; further details are given in Table 3.

Table 3 INEX Collections Test.

Collections	**Size(GB)**	$N_{document}$	**Avg. Depth**	**Objectives**
INEX-IEEE	0.70	16,819	6.90	- Ranking techniques - Compression
INEX-Wiki06	4.20	659,338	6.72	- Ranking techniques - Compression
INEX-Wiki09	50.70	2,666,190	6.72	- Ranking techniques - Compression
INEX-WS-2010	0.05	1,987	4.00	- Retrieval System
INEX-IMDB-2010	1.40	2,250,098	4.00	- Retrieval System

Methods

In the following section, we propose novel methods specifically designed for XML retrieval in accordance with the thesis's objectives.

1. Finding XML Information Granularity using the Score Sharing Algorithm

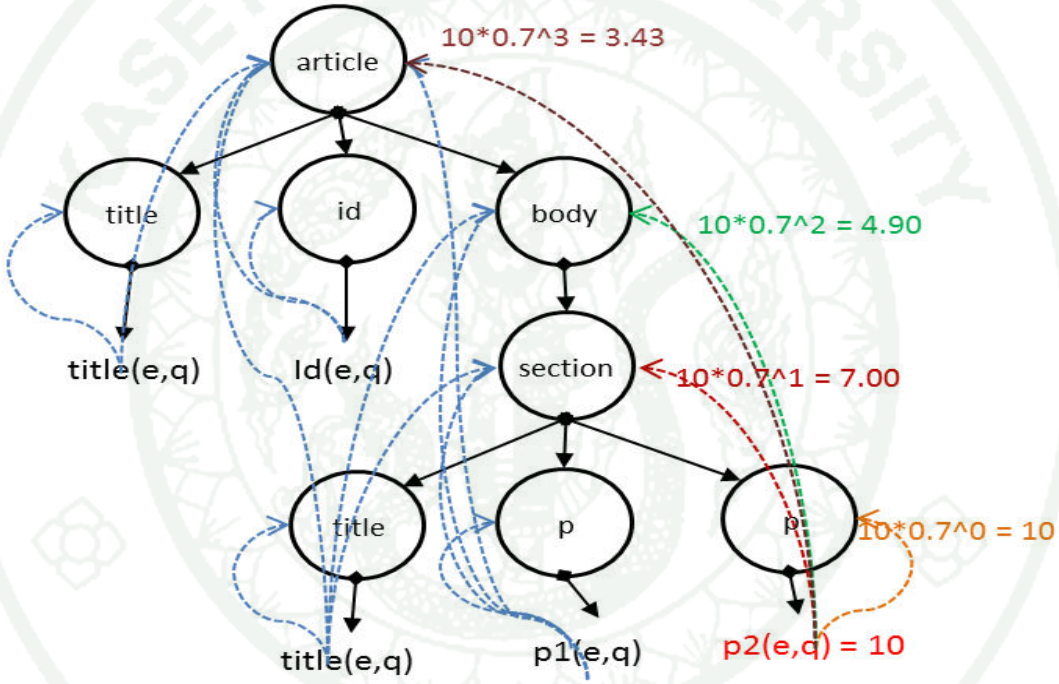


Figure 17 An illustrations of Score Sharing processing

The objective in XML retrieval is to find the appropriate element level in answering a query. Returning the smallest unit of complete information and removing the nested elements in the relevant list would be beneficial to the user. In this section, the aim is to devise a new algorithm with scoring and returning elements at the right level of granularity. To determine the appropriate level of component granularity, we proposed a novel technique that allows assigning an element a score by assigning a proportion of the leaf node scores to their element parents. This approach uses a top-down scheme, which is called the "Score Sharing function". Our approach is also

based on leaf-only indexing, because we believe the smallest leaf node contains relevant information such as the “title” element in Figure 1. In addition, leaf-only indexing can be performed automatically without any small unit issue. What is significant here is the distance between an element and its descendants, and also the owner of the content. According to classical IR weighting, we also compute the scores of all leaf elements that contain direct text by using the BM25 scoring function (Stephen *et al.*, 1994) in the collection that contains the query terms. For assigning the score for each non-leaf element, proportions of the scores from the most specific leaf elements are shared with all inner elements e (the elements do not indexed or non-leaf elements). The score of inner element e is decreased (see Figure 18) or increased (see Figure 20) by multiplying them by the power of the β smoothing parameter. In this way, more specific elements are preferred thus solving the nested elements problem during query processing time. Considering the scores of each element e by accounting for its relevant descendants e_c the details of processing are as shown in Figure 17.

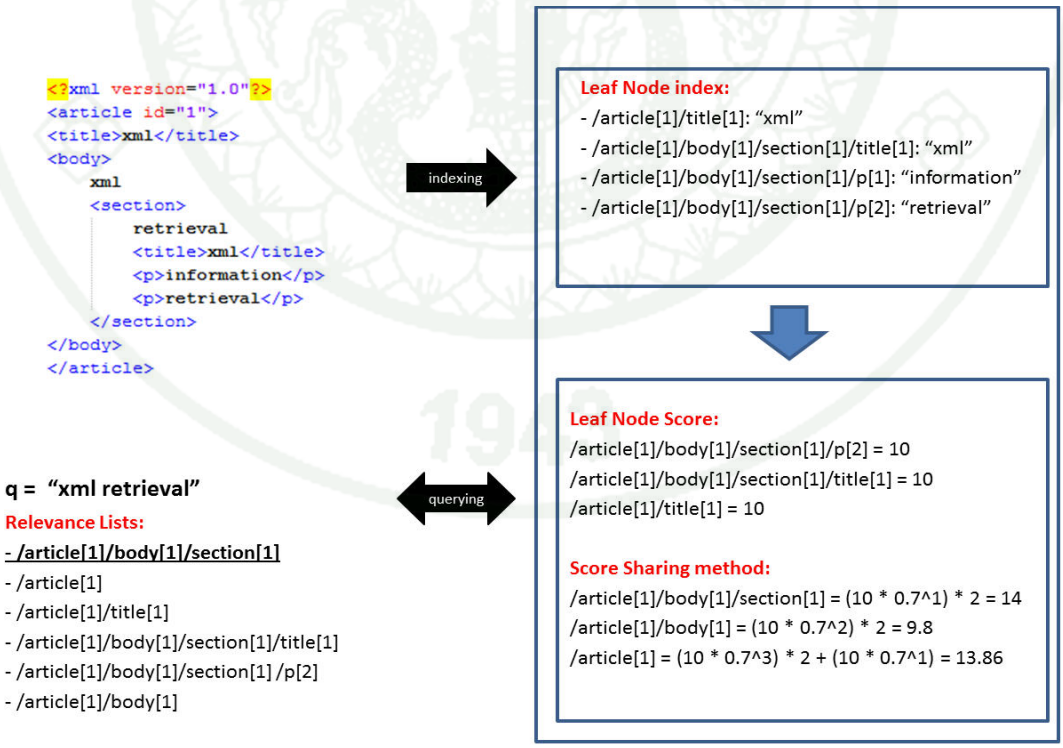


Figure 18 An example of score sharing processing given the parent node

The scores of retrieved elements $Score(e,q)$ are now shared from the child c of element e at the leaf level (“title(e,q)”, “id(e,q)”, “p1(e,q)”, and “p2(e,q)”) and their parents (“section”, “body”, and “article”) in the document XML tree as follows:

$$Score(e_c,q) = \sum_{t \in q} ief_t * \frac{(k_1+1)*tf_{t,e}}{k_1 * \left((1-b)+b*\frac{len(e)}{avel}\right)+tf_{t,e}} \tag{51}$$

$$ief_t = \frac{\log\left[\frac{N-e_t+1}{e_t}\right]}{\log(N+1)} \tag{52}$$

where

$Score(e_c,q)$ measures the relevance of child c of the element e to query q.

$tf_{t,e}$ is the frequency of term t occurring in element e.

$len(e)$ is the length of element e.

$avel$ is the average length of elements in the entire collection.

k_1 and b are used to balance the weight of term frequency and element length.

ief_t is the inverse element frequency weight of term t.

N is the total number of elements in the entire collection.

e_t is the total elements in which a term t occurs.

$$Score(e,q) \leftarrow Score(e,q) + \left(\sum_{e,c} Score(e_c,q) * \beta^n\right) \tag{53}$$

$$\beta = \begin{cases} 0.1-0.9 & \text{; if preference is given to the leaf node over the parents.} \\ \geq 1 & \text{; preference is given to the parents.} \end{cases} \tag{54}$$

where

$Score(e,q)$ is a current parent node e.

$Score(e_c, q)$ is a relevant child c of element e.

n is the distance between the current parent element e and the element's child c.

β is the smoothing parameter.

Algorithm 1 Score Sharing Algorithm

$List_{rel}$: is the relevance list
$List_{champion}$: is the Champion list
L_{ln}: is the list of relevance from the Leaf Node index
$Weight_{ln}$: is the Weight of the Leaf Node index
LN_{path}: is the absolute XPath from the Leaf Node index
$List_{rel}$: is the final list of relevance
β: is a tuning parameter (default is 0.7)
$dist$: is a distance of node

SCORESHARING(List L_{ln})
BEGIN
1: **SET** β := 0.7
2: **SET** $List_{rel}$:= NULL
3: **SET** $List_{champion}$:= NULL
4: **READ** List L_{ln}
5: **FOREACH** List LN **IN** L_{ln}
6: **SET** $dist$:= 0
7: **SET** $Weight_{ln}$:= LN_{score}
8: **FOREACH** Path LN_p **IN** LN_{path}
9: **COMPUTE** $dist$:= $LN_p.Lenght$ - $LN_p.Position$
10: **COMPUTE** $Score_{ln}$:= $Weight_{ln}$ * β^{dist}
11: **IF** $List_{champion}$ has contain Path LN_p **THEN**
12: **SET** $List_{champion}[LN_p] \leftarrow List_{champion}[LN_p]$ + $Score_{ln}$
13: **ELSE**
14: **ADD** LN_p, $Score_{ln}$ **TO** List $List_{champion}$
15: **END-IF**
16: **END-FOR**
17: **END-FOR**
19: **SET** $List_{rel} \leftarrow List_{champion}$.RemoveOverLapping()
20: **RETURN** $List_{rel}$
END.

Recalling our example and assuming that β is 0.7, then the score sharing calculations are shown in Figure 18. In this case, assume the score for each leaf node is "10", we calculate the distance *dist* as follows:

1. The root node "article[1]" contains three specific of leaf elements "article[1]/title[1]", "section[1]/title[1]", and "p2[1]". In the first traversing of the tree to text from the "article[1]" node over all the nodes: "body[1]", "section[1]", and "p[2]", the distance *dist* is 3. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^3 = 3.43$. In the second traversing of the tree to text from the "article[1]" node over all the nodes: "body[1]", "section[1]", and "section[1]/title[1]", the distance *dist* is 3. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^3 = 3.43$. In the third traversing of the tree to text from the "article[1]" node over one node, the "article[1]/title[1]" node, the distance *dist* is 1. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^1 = 7$. The total score for "article[1]" node is $(3.43 + 3.43 + 7.0) = 13.86$

2. The parent "body[1]" node contains two specific of leaf elements: "section[1]/title[1]", and "p2[1]". In the first traversing of the tree to text from the "body[1]" node over all the nodes: "section[1]", and "section[1]/title[1]", the distance *dist* is 2. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^2 = 4.90$. In the second traversing of the tree to text from the "body[1]" node over all the nodes: "section[1]", and "p[2]", the distance *dist* is 2. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^2 = 4.90$. The total score for the "body[1]" node is $(4.90 + 4.90) = 9.80$

3. The parent "section[1]" node contains two specific of leaf elements: "section[1]/title[1]", and "p2[1]". In the first traversing of the tree to text from the "section[1]" node over the node "section[1]/title[1]", the distance *dist* is 1. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^1 = 7.0$. In the second traversing of the tree to text from the "section[1]" node over the node "p[2]", the distance *dist* is 1. The score is decreased with respect to the distance *dist*, so then the score is $10 * 0.7^1 = 7.0$. The total score for the "section[1]" node is $(7.0 + 7.0) = 14.0$

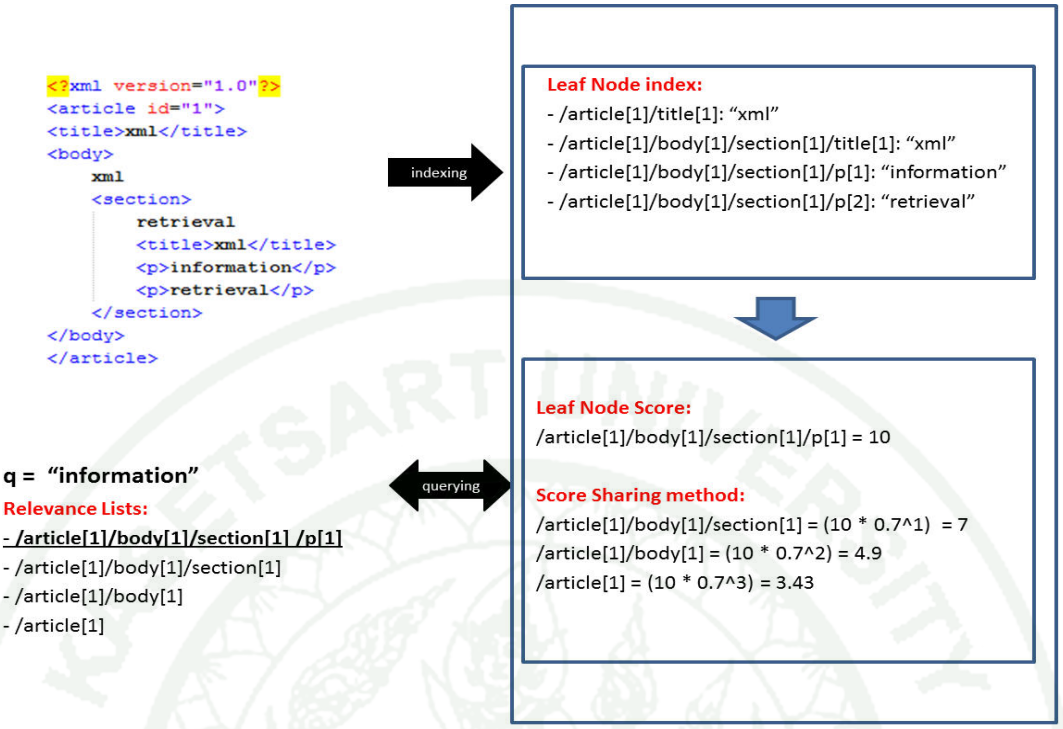


Figure 19 An example of score sharing processing given the leaf node

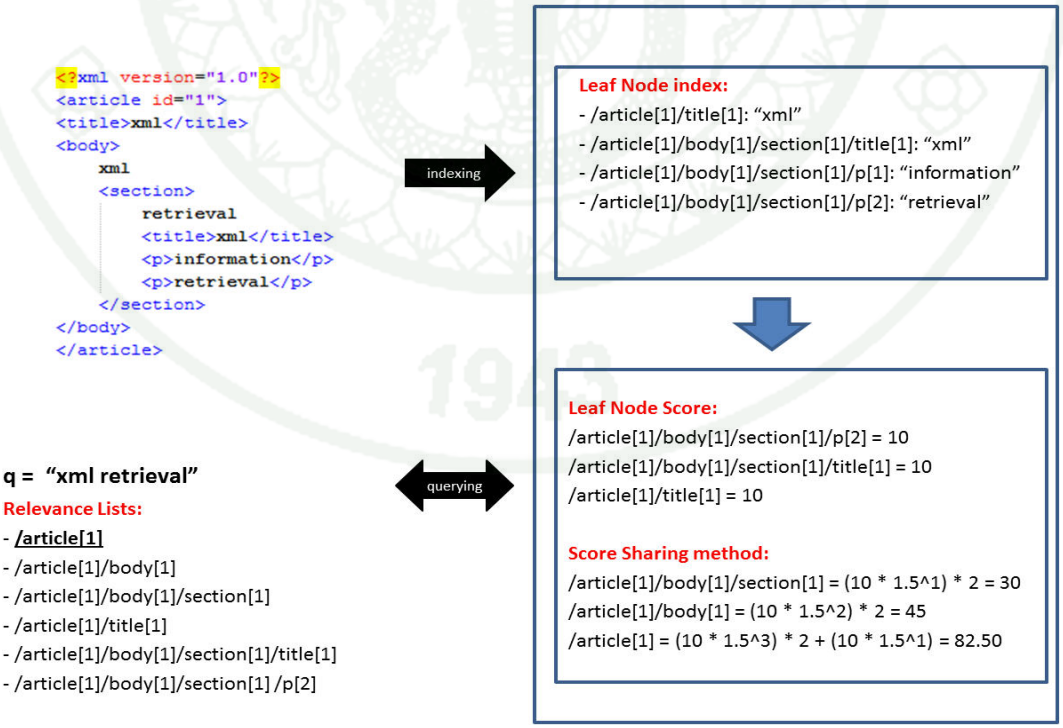


Figure 20 An example of score sharing processing given the root node

Alternatively, we can make a recommendation for β under the score sharing algorithm when a user prefers to implement the algorithm at the root level. This feature of the algorithm can be activated by setting the β parameter to a value greater than 1. To illustrate, assume β is 1.5, then the score for each parent is computed as shown in Figure 20.

Algorithm 2 AutoMix Algorithm

N: is a Node in document D
C_n: is a Child node of N
T: is a Text Node
SW_{index}: is a Selected index
LN_{index}: is a Leaf Node index

SELECTMIXCONTENT(Node N)
BEGIN
1: **READ** Node N in document D
2: **FOREACH** Node N **IN** D
3: **IF** N contains Child C_n **THEN**
4: **IF** N contains Text T **THEN**
5: **ADD** Node N and Text T **TO** SW_{index}
6: **END-IF**
7: **FOREACH** Child C_n **IN** Node N
8: SELECTMIXCONTENT(C_n)
9: **END-FOR**
10: **ELSE**
11: **IF** N contains Text T **THEN**
12: **ADD** Node N and Text T **TO** LN_{index}
13: **END-IF**
14: **END-IF**
15: **RETURN** SW_{index} and LN_{index}
END.

2. Ranking with the Double Scoring Algorithm

The selected type approach is the XML ranking strategy that employs a selected element list and ranks the element with the most relevance for return to users. However, this strategy raises the issue of manually choosing and assigning weights to all potentially retrievable elements. In order to solve this issue, we briefly describe two automatic methods. The first method can be used for automatic tuning of fields using mixed content elements. The second method implements the scoring function to assign weights to each selected field obtained by the first method. The details are as follows:

2.1 Automatic Tuning with Mixed Content Element

To solve the first problem related to BM25F (34), i.e. automatic field selection, a new automatic tuning is proposed. Our approach considers the use of an automatic method to tune selected fields using only mixed content elements. We believe that mixed content elements can help in deriving an automatic method for tuning selected fields. Therefore, the aim is to investigate the way in which *mixed content* can be used to identify highly relevant sections. We propose the use of a method called Automatic Tuning of Mixed Element (AutoMix) for the tuning of mixed content elements. The mixed content elements are separated into new indices known as Selected Weight (SW).

Referring to the example of an XML element tree, we select the mixed content element and then we build the Selected Wight index as shown in Figure 21.

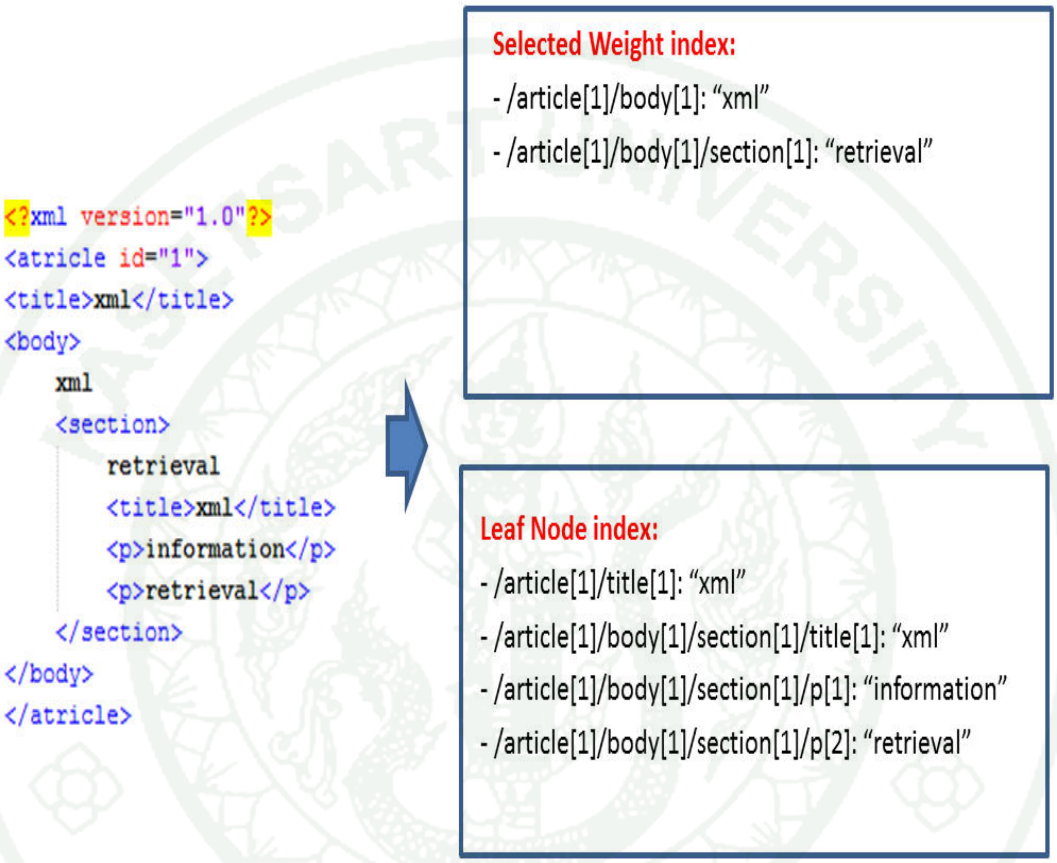


Figure 21 An example of AutoMix processing

2.2 Double Scoring Function

In this section, we present a new technique to reduce parameter tuning by using a new extended index that stores all of the mixed-content nodes. Selected indexing is similar to traditional information retrieval because each XML node is a bag of words and can be scored similarly to ordinary plain text documents, and then the Selected Weight index can be calculated using as a baseline of BM25 function (Stephen *et al.*, 1994).

According to the significance of the mixed-content node, we have to reward the score for each leaf node that contains the path of mixed-content, and this approach multiplies the weight of the relevant lists by the weight form the Selected Weight list which we call the Double Scoring algorithm. The details of processing are as shown in Figure 22.

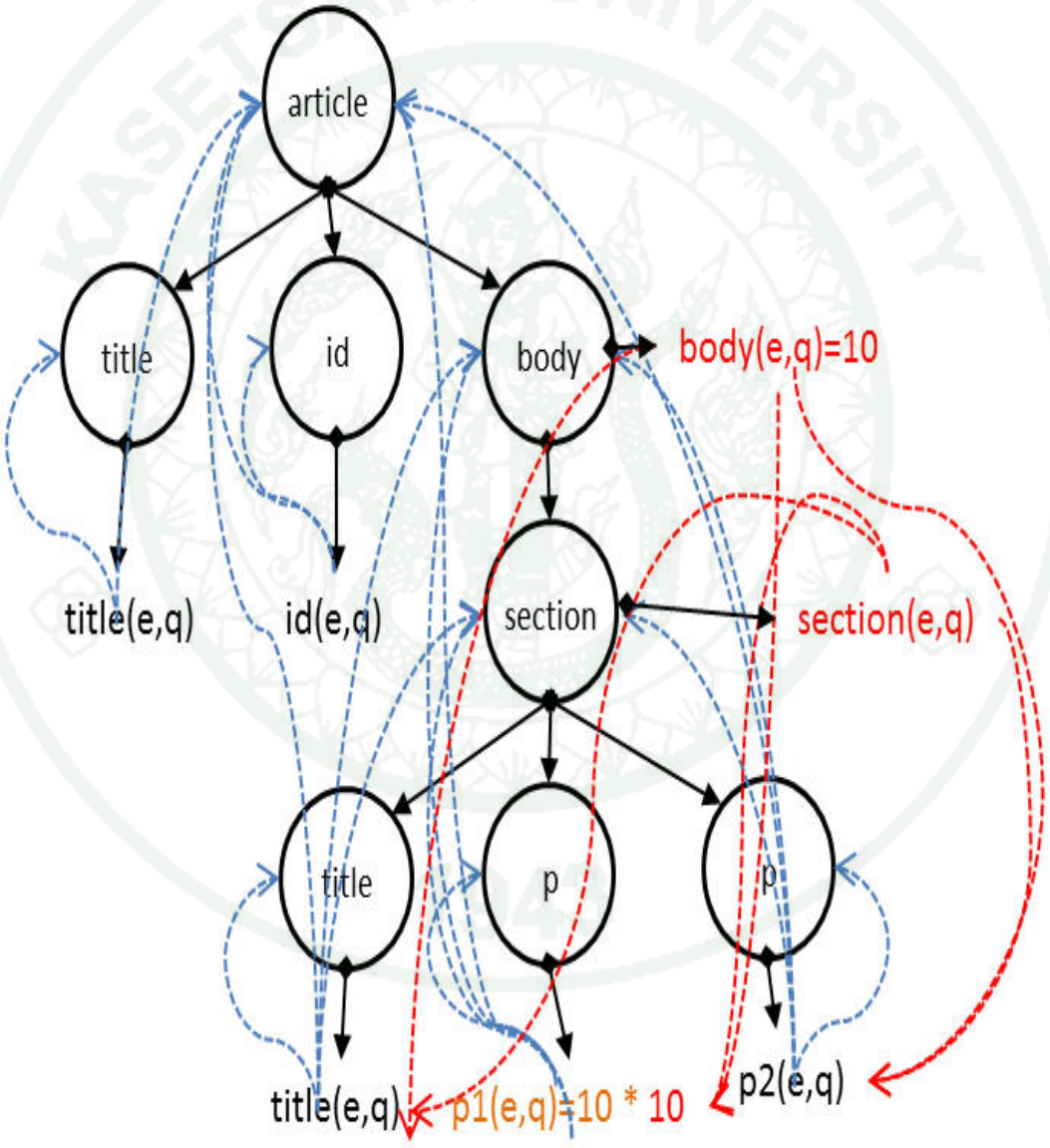


Figure 22 An illustration of Double Scoring processing

Afterwards, we can compute the element score as follows:

Algorithm 3 DoubleScoring Algorithm

L_{rel}: is the relevance list
L_{sw}: is the list of relevance from the Selected Weight index
L_{ln}: is the list of relevance from the Leaf Node index
$Weight_{sw}$: is the Weight of the Selected Weight index
$Weight_{ln}$: is the Weight of the Leaf Node index

DOUBLESCORING(List L_{sw}, List L_{ln})
BEGIN
1: **READ** List L_{sw}, List L_{ln}
2: **SET** L_{rel} := NULL
3: **FOREACH** List SW **IN** L_{sw}
4: **SET** $Weight_{sw}$:= SW_{score}
5: **FOREACH** List LN **IN** L_{ln}
6: **SET** $Weight_{ln}$:= LN_{score}
7: **IF** $SW \in LN$ **THEN**
8: **ADD** LN, $Weight_{sw}$ * $Weight_{ln}$ **TO** List L_{rel}
9: **ELSE**
10: **ADD** LN, $Weight_{ln}$ **TO** List L_{rel}
11: **END-IF**
12: **END-FOR**
13: **END-FOR**
14: **RETURN** $List_{rel}$
END.

From field weights W_f function of the BM25F scoring according to (Stephen *et al.*, 2004) as follows:

$$tf_{d,t,f} = \sum_{f} W_f * tf_{d,t,f} \tag{55}$$

We can see that a linear combination of the weighted field frequencies is used instead of the original term frequency in specified fields. We hypothesize that this method can also be applied to all of the mixed content elements that we propose in the Selected Weight index.

Suppose, we have n mixed content nodes in a given collection C. Given a weight $W_{f,n}$ for each element n in the Selected Weight index. These index is similar to traditional information retrieval because each mixed content node is a bag of words of itself. Therefore, the BM25 scoring function cannot benefit from the information contained in the fields with less text (Stephen and Hugo, 2009). Thus, we separate the mixed content elements in the result set while keeping elements under the same index of leaf-node content, and then we calculate the Selected Weight using the term frequencies according to (Trotman, 2005). For each field weight W_f of BM25F, we calculate the weight for each relevance list of Selected Weight $W_{f,n}$, instead of tuning each selected field as follows:

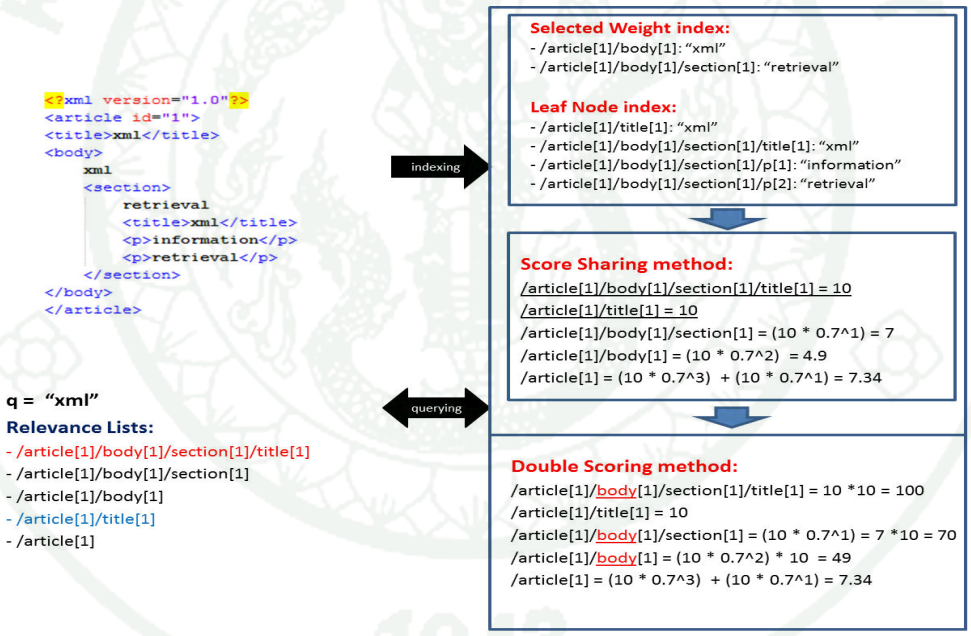


Figure 23 An illustration of query processing with Double Scoring

$$W_{f,n} = SW(e,q) \tag{56}$$

where

$W_{f,n}$ is the field weight of mixed-content elements e in the Selected Weight index.

And then, we compute score for each element e as follows:

$$SW(e,q) = \sum_{t \in q} tf_e \tag{57}$$

where

$SW(e,q)$ measures the relevance of element e in the Selected Weight index to query q. tf_e is the frequency of term t occurring in element e.

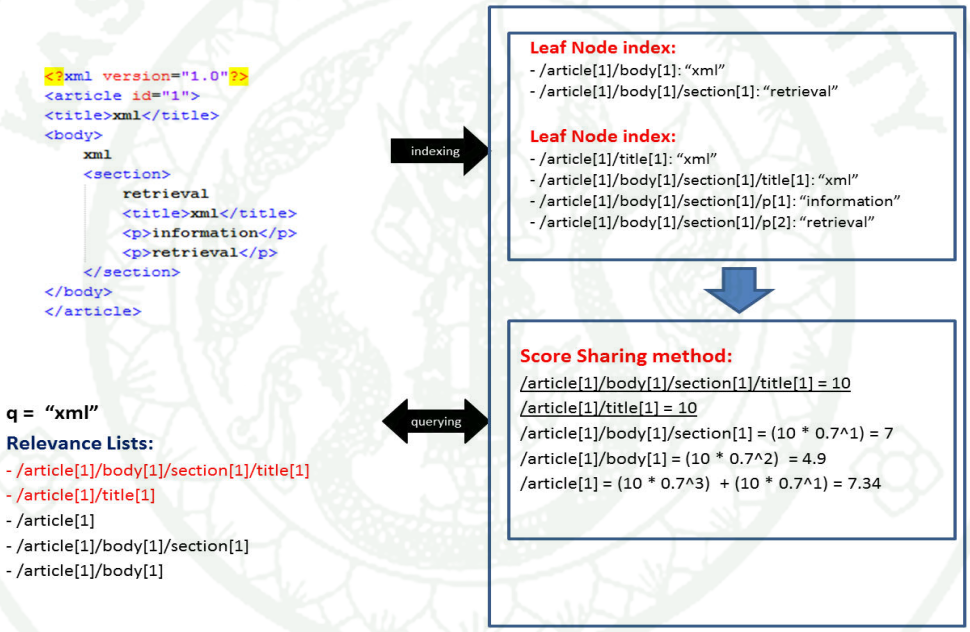


Figure 24 An illustration of query processing without Double Scoring

We run the query q in parallel on each index (i.e., the Selected Weight index and Leaf-Node index), and then we use the new score fromr each list to implement the BM25W as follows:

$$BM25W(e,q) = Score(e,q) * \prod_{i \in n} SW(e_i,q) \tag{58}$$

where

The $BM25W(e,q)$ measures the relevance of element e to query q.

For instance, the query q contains "xml retrieval", consider the LN_{index} and SW_{index} indices, we assume the score $Score(e,q)$ for each element e is 10 and β is 0.7, the calculations are as shown in Figure 24 and Figure 23. As a result, the Double Scoring function solves the issue of manually choosing and assigning weights to all potentially retrievable elements. Our approach selects mixed-content elements and automatically assigns weights at query time, and also does not require training data or an evaluation set.

Following this, we compare the double scoring (BM25W) with the BM25F function as shown in Table 4.

Table 4 Comparison of the BM25W and the BM25F functions.

BM25W	**BM25F**
1. This model has two indices, both an SW and a Leaf Node index	1. This model has only the Leaf Node index
2. The document fields are automatically chosen using mixed content elements	2. The document fields are selected manually by an expert
3. W_f is the weight automatically assigned at query time	3. W_f is the weight manually assigned in data pre-processing time
4. W_f does not require a tuning parameter	4. W_f requires a tuning parameter
5. W_f does not require training data and an evaluation set	5. W_f requires training data and an evaluation set

3. Optimizing query processing with the Compression Technique

Even though XML structure can be beneficial, it can be time and space consuming as well and also influences the efficiency of the search system (Anh and Moffat, 2002; Fuhr and Gövert, 2002).

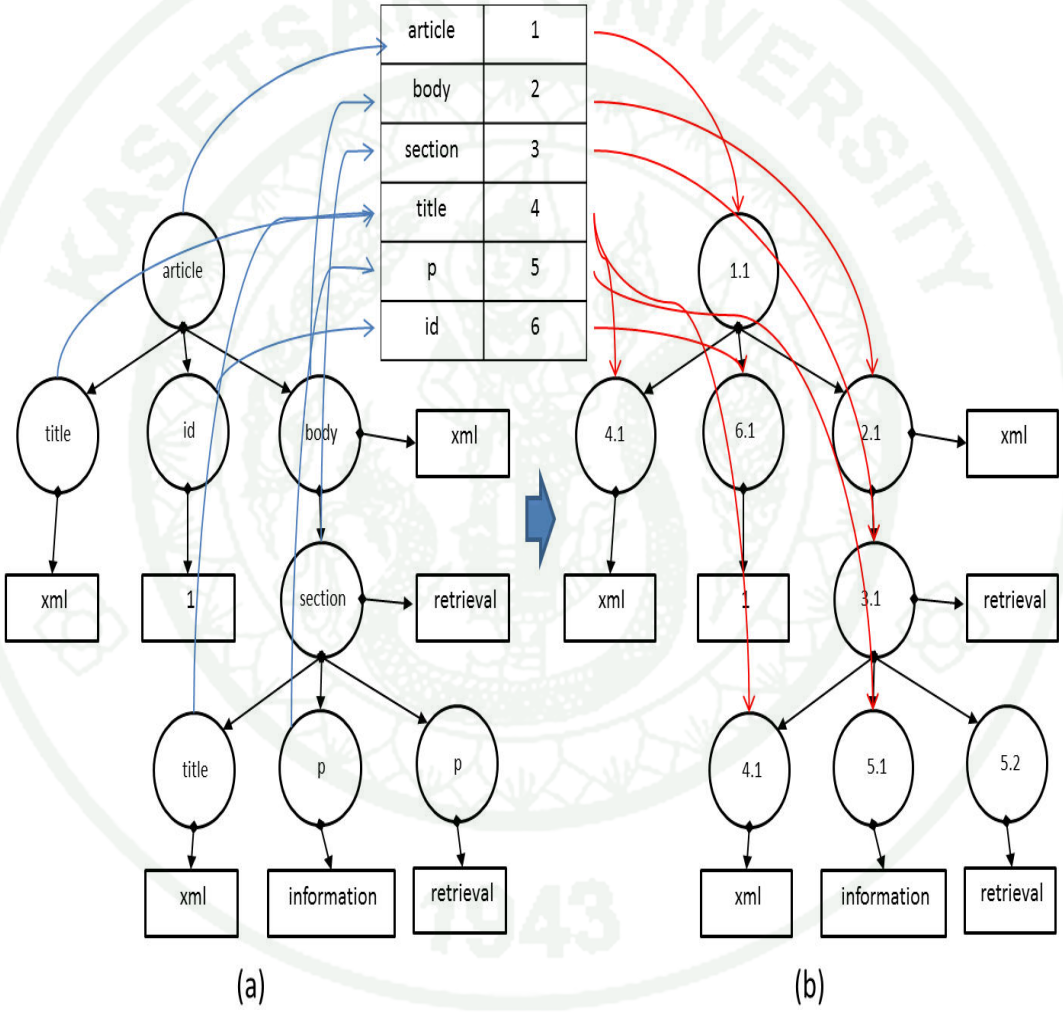


Figure 25 An example of a Compressed XML Element Tree

The main disadvantage of using XML documents is that they are large in size because they are highly structured and often have long tag and attribute names, for instance, the element name in Wikipedia; “management-note[1]”, “broadcasting-station[1]”, “system-of-measurement[1]”. In XML retrieval systems, the frequency of

tag occurrences and the position of tags are very important. It is possible to modify the queriable type compression algorithms to handle this issue.

Algorithm 4 Compression Algorithm

L_{ln}: is a relevance of Leaf Node
$L_{champion}$: is a Champion list
P_{name}: is the name of node P
$P_{position}$: is the position of node P
L_{path}: is the list of XPath
L_{final}: is a final list
ecADXPIcompressor(List L_{ln})
BEGIN
1: **READ** List L_{ln}
2: **FOREACH** List L **IN** L_{ln}
3: **SET** L_{final} := NULL
4: **SET** $L_{champion}$:= NULL
5: **FOREACH** Path P **IN** List L_{path}
6: **IF** $P_{name} \in L_{champion}$ **THEN**
7: **SET** P_{name}, $L_{champion}$ [P_{name}] + 1 **TO** List $L_{champion}$
8: **ELSE**
9: **ADD** P_{name}, 1 **TO** List$L_{champion}$
10: **END-IF**
11: **END-FOR**
12: **END-FOR**
13: **SET** $L_{champion} \leftarrow L_{champion}$.SortByValue()
14: **SET** $Counter$:= 0
15: **FOREACH** Path P **IN** List $L_{champion}$
16: **IF** $P \in L_{final}$ **THEN**
17: **ADD** P_{name}, $Counter.P_{position}$ **TO** ListL_{final}
18: **ELSE**
19: **COMPUTE** $Counter \leftarrow Counter + 1$
20: **ADD** P_{name}, $Counter.P_{position}$ **TO** ListL_{final}
21: **END-IF**
22: **END-FOR**
23:
24: **RETURN** List L_{final}
END.

In this work, we enhanced existing compressors which use a Dictionary Based algorithm in the devising what we call the extension XML compression of ADXPI. It is based on the principle of extracting XPath and position from the document, and grouping them based on the name of tag. The document is encoded as a sequence of

integers, while the data grouping is based on XML tags and the position of the tags is mapped to the decimal value of tag mapping using integer values.

Therefore, compressing XML enhances both efficiency and convenience. The re-organized data is now compressed by an adaptive Dictionary Based technique; it has extremely accurate compression, and it eliminates the repetition of the dictionary based words in the database. We derive probabilities, which dynamically change with the frequency of tag name and this allows us to directly retrieve the path in the compressed data. Since the data volume is reduced, such compressing of data path may allow even faster retrieval than in the original data.

Recalling our example in Figure 1, compression algorithm processing of all leaf nodes is as follows:

1. article[1]/title[1]: “xml”
2. article[1]/@id[1]: “1”
3. article[1]/body[1]: “xml”
4. article[1]/body[1]/section[1]: “retrieval”
5. article[1]/body[1]/section[1]/title[1]: “xml”
6. article[1]/body[1]/section[1]/p[1]: “information”
7. article[1]/body[1]/section[1]/p[2]: “retrieval”

We can split all leaf nodes and construct the Champion list as follows:

1. 1st tag is “article”, frequency is 7.
2. 2nd tag is “title”, frequency is 2.
3. 3rd tag is “body”, frequency is 5.
4. 4th tag is “section”, frequency is 4.

5. 5th tag is “p”, frequency is 2.

6. 6th tag is “@id”, frequency is 1.

Table 5 Details of Leaf Node Data.

Element-ID	Element	Context
1	x1/1.1/4.1	xml
2	x1/1.1/6.1	1
3	x1/1.1/2.1	xml
4	x1/1.1/2.1/3.1	retrieval
5	x1/1.1/2.1/3.1/4.1	xml
6	x1/1.1/2.1/3.1/5.1	information
7	x1/1.1/2.1/3.1/5.2	retrieval

Table 6 Details of Mapping Data.

MapID	MapPath	MapPosition
1	article[1]	1
2	body[1]	1
3	section[1]	1
4	title[1]	1
5	p[1]	1
5	p[2]	2
6	@id[1]	1

Following the result list as above, we sort the Champion list by frequency to obtain the Final list and map as follows:

1. 1st tag is “article[1]”, frequency is 7 and then the mapping is 1.1.

2. 2nd tag is “body[1]”, frequency is 5 and then the mapping is 2.1.

3. 3rd tag is “section[1]”, frequency is 4 and then the mapping is 3.1.

4. 4th tag is “title[1]”, frequency is 2 and then the mapping is 4.1.

5. 5th tag is “p[1]”, frequency is 2 and then the mapping is 5.1.

6. 6th tag is “p[2]”, frequency is 2 and then the mapping is 5.2.

7. 7th tag is “@id[1]”, frequency is 1 and then the mapping is 6.1.

As a result, we have the XML element tree shown in Figure 25(a), and the compressed of XML element tree shown in Figure 25(b). Finally, the data of the leaf-node index is stored with details as shown in Tables 5 and 6.

3.1 Support for Content and Structure Queries

We enhance the compression to support content-and-structure queries, which are elements that have been found to contain at least one instance of a term that was a specified path in the filter (Sigurbjörnsson *et al.*, 2004; Kamps *et al.*, 2005b). For instance, the following information need “retrieve document sections with paragraphs contain the word *information*”

//section[about(//p, “information”)]

In the first step, we will transpose the query to the compressed data with respect to Table 5 as follows:

//3[about(//5, “information”)]

The first filter looks for occurrences of the term “information” in elements whose context matches the path “**//3//5**”. If we find that the term “information” occurs in an element with the context “/1.1/4.1” this is not a valid support for this filter. However, if we find a single occurrence of “information” in the context of path “/1.1/2.1/**3**.1/**5**.1” this would be a valid support. Once we have removed all supports that do not represent valid supports (according to the filter) then we can create the return elements for this filter. In this case the return path is the “/1.1/2.1/**3**.1/**5**.1” and having one hit for the term “information”. It is possible to assign more weight for a return element that contains more than one supported content-and-structure constraint.

4. An Implementation for XML Retrieval System

Finally, we demonstrate XML retrieval, namely the More Efficient XML Information Retrieval (MEXIR). This system is based on a leaf-node indexing scheme that uses a relational DBMS as a storage back-end. We discuss the schema setup using MySQL and the full-text engine Sphinx with the MySQL dump function.

Sphinx has two types of weighting functions:

1. The phrase rank is based on the length of the longest common subsequence of search words between the document body and query phrases. If there is a perfect phrase match in some document, then its phrase rank would be equal to the number of query words, which would be the highest possible rank.

2. The statistical rank is based on the classic BM25 function, which only takes word frequencies into account. If a word is rare in the entire database, it receives more weight.

At first, we consider a simplified XML data model, but we disregard Meta mark-up such as comments, links and attributes. Figure 26 depicts an overview of the XML retrieval system. The main components of the MEXIR retrieval system are as follows:

1. When new documents are entered into the system, the Absolute Document XPath Indexing (ADXPI) (Wichaiwong and Jaruskulchai, 2011d) indexer parses and analyzes the name of each element and its position to build inverted lists for each index in this system.

2. The extension XML compression for ADXPI (ecADXPI) compressor analyzes the frequency of each element and its position to build the mapping of a dictionary base, which is stored as compressed data in the MySQL database.

3. The AutoMix algorithm analyzes the tree position for each element to separate content to build the Selected Weight (SW) and Leaf Node indices, which are stored in the MySQL database.

4. The SphinxDB search engine is used to build both indices in the system. The Selected Weight index is based on Term Frequency, and the Leaf Node index is based on the classic BM25 function.

5. The Score Sharing function is used to assign parent scores by assigning a proportion of the scores of the leaf nodes to their parents using a top-down approach.

6. The Double Scoring function is used to adjust the Leaf Node scores based on linear combination.

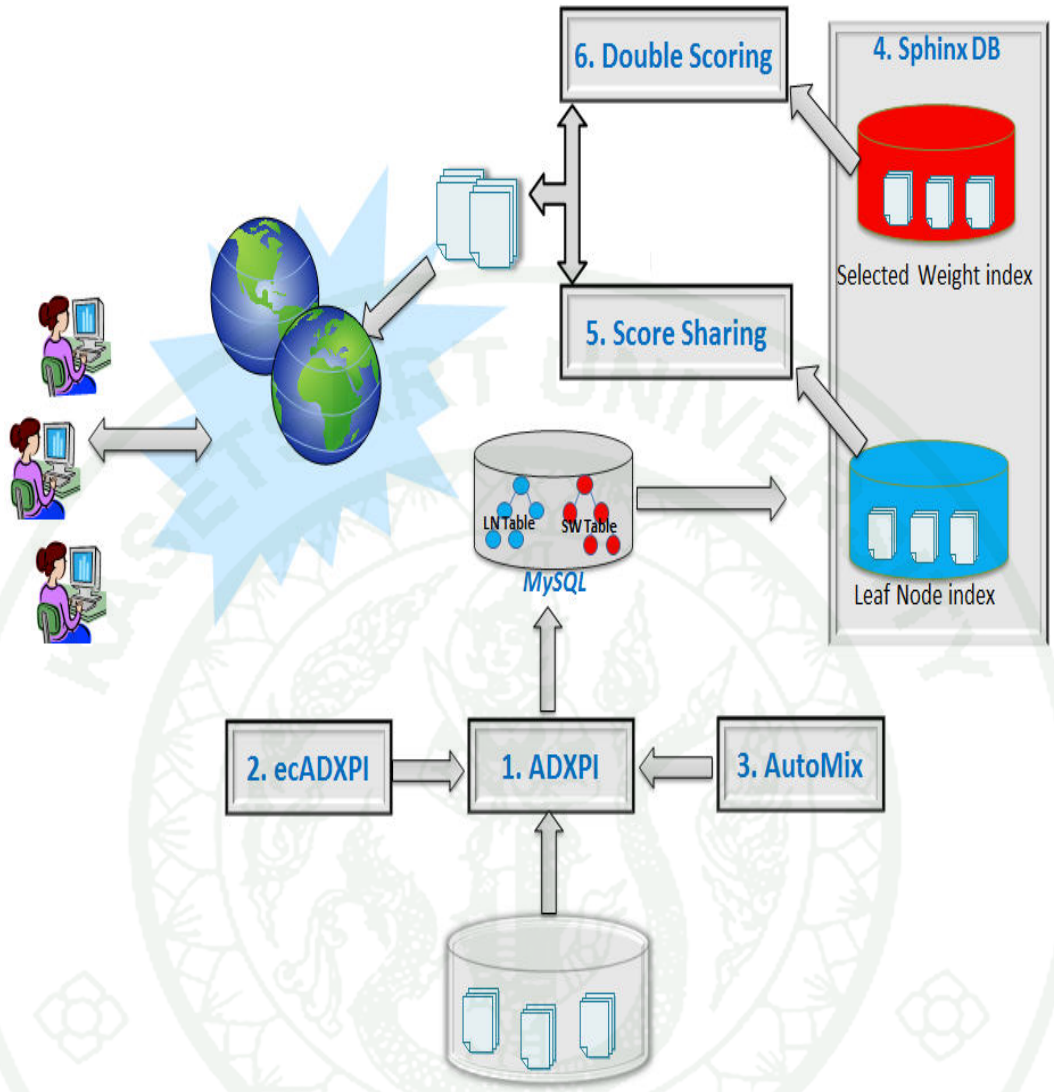


Figure 26 An overview of the MEXIR System

RESULTS AND DISCUSSION

1. Evaluation Methods

Evaluating the effectiveness of ranking techniques is very important in information retrieval. The two main objectives of an information retrieval system are coverage and accuracy. Standard XML measurements are used to evaluate different aspects of the Scoring Sharing and the Double Scoring algorithms such as nxCG, iP and MAiP.

Table 7 The Effectiveness on INEX-IEEE of Focused Task at nxCG (OVERLAP=on, QUANT=gen).

RUN ID	nxCG@5	nxCG@10	nxCG@25	nxCG@50
TF-IDF-SS	0.1971	0.1400	0.1123	0.0942
TF-IDF	0.1667	0.1342	0.1026	0.0818
%	**18.24**	**4.32**	**9.45**	**15.16**

In the data compression experiment, the effectiveness in data compression is the proportion of compression, which can be found as follows:

$$Size_{ratio} = \left[1 - \frac{Size_{compress}}{Size_{actual}}\right] \tag{59}$$

And the effectiveness of response time is the proportion which can be found as follows:

$$Time_{ratio} = \left[1 - \frac{Time_{compress}}{Time_{actual}}\right] \tag{60}$$

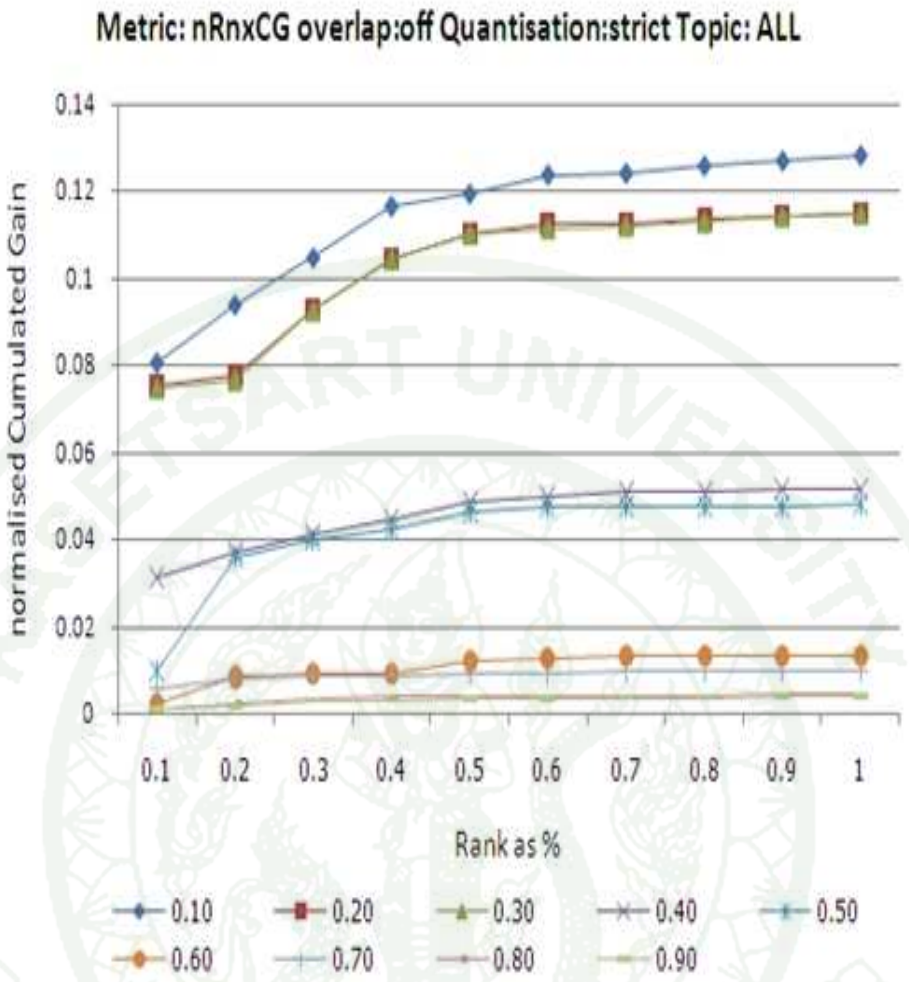


Figure 27 Variation in the value of the β parameter

2. Best Ranking Results and Discussion

2.1 Information Granularity Results and Discussion

In this section, we address the issue of XML-IR to improve the effectiveness of the search system. The system will try to discard some small units of information and nested elements by collapsing the relevance results in order to find the optimum element granularity to return to users. We report the results of evaluation of the score sharing algorithm in terms of nxCG, iP[0.01] and MAiP.

We first tuned the β parameter using the INEX-2005 Adhoc track evaluation scripts distributed by the INEX. Our tuning approach has shown that the sums of all relevance scores are maximized as shown in Figure 28 and Figure 27, and then the β parameter is set to 0.10, which is used to compute the combination function after scoring the leaf elements.

In the following section, we report the effectiveness of the scoring sharing function on INEX collections as follows:

Table 8 The Effectiveness on the INEX-Wiki06 Focused Task.

INEX	RUN ID	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
2006	TF-IDF-SS	0.4518	0.4044	0.3651	0.1109
	TF-IDF	0.4136	0.3468	0.1962	0.0796
	%	**9.24**	**16.61**	**86.09**	**39.32**
2007	TF-IDF-SS	0.4169	0.3186	0.2539	0.0987
	TF-IDF	0.3592	0.3096	0.1758	0.0768
	%	**16.06**	**2.91**	**44.43**	**28.52**
2008	TF-IDF-SS	0.5740	0.4262	0.3411	0.1187
	TF-IDF	0.4526	0.3668	0.3155	0.0847
	%	**26.82**	**16.19**	**8.11**	**40.14**

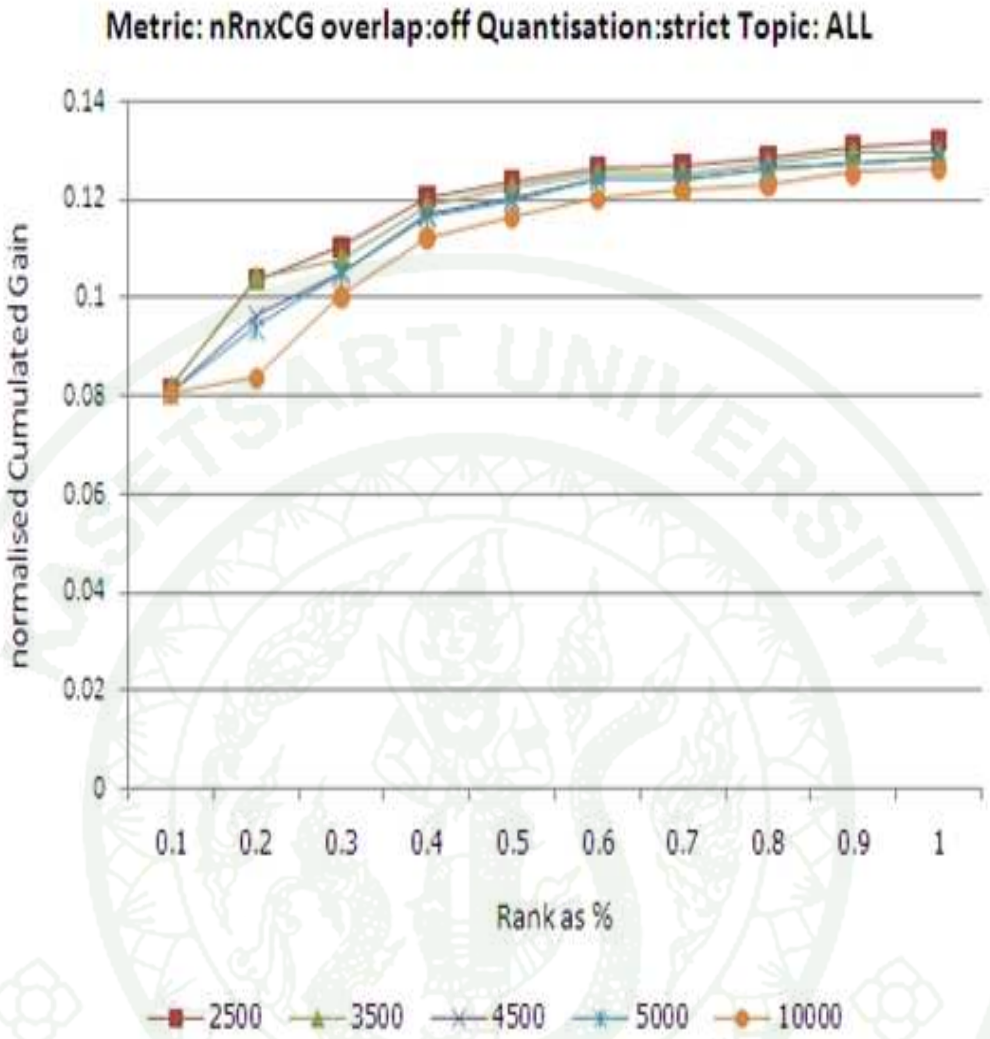


Figure 28 Variation in the total number of leaf elements

To start with, the sensitivity of the distribution of the appropriate level of granularity depends on the total number of elements in the collection with respect to the Score Sharing function. In Table 10, we report our combination approach showing the collapsing of elements up to 14.69%. Our experiment shows that the element “bm” or “back matters” is the most frequent document part, and the appropriate level is the second level with respect to the average depth of a node in the collection, which is 6.90 on the INEX-IEEE as shown in Table 9. For the INEX-Wiki06, the element “p” or “paragraph” is the most frequent document part, and the appropriate level is the third

level with respect to the average depth of a node in the collection, which is 6.72 as shown in Table 9 and Table 10. For the INEX-Wiki09, the element "p" or "paragraph" is the most frequent document part, and the appropriate level is the fifth level with respect to the average depth of a node in the collection, which is 6.72 as shown in Table 9 and Table 10.

Table 9 Distribution of Element Relevance.

Collections	**Element**	N_{freq}	**%**
INEX-IEEE	bm (back matters)	4,406	47.73
	p	3,653	39.57
	ip1 (initial paragraph)	772	8.36
	ss1 (subsection level 1)	209	2.26
	fm (front matter)	115	1.25
	ss2 (subsection level 2)	33	0.36
	p2	33	0.36
	kwd (Keyword)	5	0.05
	p1	5	0.05
INEX-Wiki06	p	311,292	46.53
	section	222,628	33.28
	body	132,857	19.86
	gallery	2,068	0.31
	sub	178	0.03
INEX-Wiki09	p	514,396	97.46
	company	4,410	0.84
	terminal	4,167	0.79
	railway-station	3,056	0.58
	person	1,798	0.34

For comparison of effectiveness, we present in Table 7 the results in terms of the nxCG metric at early ranks, the official measure of the INEX-IEEE collection. The results show that using the score sharing approach achieved an improvment of nxCG@5 over the baseline TF-IDF of up to 18.24%. In Table 8, we report the interpolated precision at difference recall levels and MAiP computed for 101 recall levels. For this experiment, the score sharing was also applied to the baseline TF-IDF for iP[0.01] and MAiP, the improvements were 26.82% and 40.14%, respectively.

In this analysis, we take the results that were obtained from the TF-IDF-SS, in which proportions of the leaf scores are assigned to their parent elements

and compare them with the results from the baseline TF-IDF. This shows again that TF-IDF-SS works well with the score sharing method on document-centric XML documents. We can conclude that significant improvement of results of the baseline TF-IDF function can be obtained from the Score Sharing method. This finding suggests that it is possible to improve the baseline TF-IDF approach, which is the usual benchmark in the INEX. The results confirm the effectiveness of the Score Sharing algorithm, and also confirm the analysis of the effect of granularity, because smaller elements are returned instead of those of article level (Pharo, 2008). The main conclusion that can be drawn from the experiments is that the Scoring sharing function is successful in automatically returning the appropriate level of an XML document.

Table 10 Distribution of Levels of Elements with and without Score Sharing.

Collections	$Depth_{SS}$	N_{freq}	$Depth_{NSS}$	N_{freq}	**%**
INEX-Wiki06	2	78325	2	67313	14.06
	3	154458	3	158855	(2.85)
	4	106157	4	111371	(4.91)
	5	31633	5	33069	(4.54)
	6	3488	6	3758	(7.74)
INEX-Wiki09	2	89971	2	76752	14.69
	3	173061	3	179179	(3.54)
	4	109482	4	114473	(4.56)
	5	30468	5	32655	(7.18)
	6	3106	6	3241	(4.35)

2.2 Double Scoring Results and Discussion

In this section, we considered the data structure of the INEX collections by using the AutoMix function, and we then constructed both Selected Weight and Leaf-Node indices and the total number of elements for each INEX collection under analysis. As seen in the experimental results shown in Table 11, the INEX-IEEE exhibited 8.59% mixed content, and the average length of the mixed content was 337.05 bytes per element. The INEX-Wiki06 exhibited 32.64% mixed content elements, and the average length of the mixed context was 15.39 bytes per element. The INEX-Wiki09 exhibited 23.63% mixed content elements with the average length of the mixed context being 52.83 bytes per element.

Table 11 Distribution of the Number of Elements in the INEX Collections.

Collections	Mixed Node	Leaf Node	Avg. Size (Byte)	%
INEX-IEEE	100,795	1,073,179	337.05	8.59
INEX-Wiki06	2,201,529	4,544,086	15.39	32.64
INEX-Wiki09	4,117,364	11,605,874	52.83	23.63

Table 12 The Effectiveness of the INEX-2008 Focused Task at iP[0.01].

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.2215	0.2419	0.2386	0.2386	0.2170
ANY	0.2189	0.4419	0.4386	0.4386	0.4170
ALL	0.2350	0.4840	0.4751	0.4751	0.4768
PHRASE	0.2230	0.4595	0.4544	0.4544	0.4514
EXTEND	0.4419	**0.6499**	**0.6499**	**0.6499**	0.5678

Table 13 The Effectiveness of the INEX-2008 Focused Task at MAiP.

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.0457	0.0561	0.0560	0.0560	0.0507
ANY	0.0401	0.0961	0.0960	0.0960	0.0907
ALL	0.0521	0.0854	0.0835	0.0835	0.0829
PHRASE	0.0459	0.0870	0.0868	0.0868	0.0868
EXTEND	0.0559	**0.1828**	0.1827	0.1827	0.1631

For the evaluation of the double scoring algorithm, we first used the Sphinx parameters for the BM25 where $k_1 = 1.20$ and $b = 0.00$ (Wichaiwong and Jaruskulchai, 2011a). To evaluate the sensitivity of the evaluation, we used the entire Sphinx match mode values for each index, including MATCH BOOLEAN (BOOLEAN–the final weight matches query as a *Boolean* expression), MATCH ANY (ANY–the final weight is a sum of weighted phrase ranks for matching *any* of the query words), MATCH ALL (ALL–the final weight is the sum of weighted phrase ranks for matching *all* query words), MATCH PHRASE (PHRASE–the final weight is the sum of weighted phrase ranks for matching the query *phrase*, which requires a perfect match), and MATCH EXTENDED (EXTEND–the final weight is the sum of weighted phrase ranks and the *BM*25 weight, multiplied by a thousand and rounded to the nearest integer). In the experiment, the columns indicate the Leaf-Node index, and the rows indicate the Selected Weight index. Thus, we report the effectiveness of our system for the INEX collections as follows:

Table 14 The Effectiveness of the INEX-2010 Focused Task at iP[0.01].

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.2011	0.2162	0.1386	0.1386	0.1170
ANY	0.2179	0.3285	0.3144	0.3284	0.2791
ALL	0.1775	0.2469	0.2432	0.2468	0.2463
PHRASE	0.1719	0.2262	0.2261	0.2261	0.2256
EXTEND	0.2198	**0.3909**	**0.3909**	**0.3909**	0.3769

Table 15 The Effectiveness of the INEX-2010 Focused Task at MAiP.

MODE	BOOLEAN	ANY	ALL	PHRASE	EXTEND
BOOLEAN	0.0205	0.0211	0.0211	0.0211	0.0199
ANY	0.0421	0.0619	0.0629	0.0624	0.0615
ALL	0.0319	0.0500	0.0535	0.0500	0.0499
PHRASE	0.0327	0.0570	0.0570	0.0570	0.0570
EXTEND	0.0455	**0.0750**	0.0749	0.0749	0.0728

Next, the performances of different Sphinx search features were evaluated. Tables 12 and 13 show the results obtained from the BM25W ranking functions on the INEX-Wiki06, and Tables 14 and 15 show the results obtained from the BM25W

ranking functions on the INEX-Wiki09.

Table 16 The Effectiveness of the INEX-2008 Focused Task.

RUN ID	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
BM25	**0.5134**	0.4790	0.4328	0.1476
BM25F	**0.5195**	0.5014	0.4657	0.1574
BM25W	**0.6058**	0.5288	0.5017	0.1728
BM25F-SS	**0.5678**	0.5324	0.4938	0.1631
BM25W-SS	**0.6499**	0.5724	0.5438	0.1828

Tables 12 and 13 show the BM25W obtained the highest scores for the INEX-Wiki06 on 2008 topics for the MATCH EXTENDED mode on the Leaf-Node index and the MATCH ANY mode on the Selected Weight index, with 0.6499 for iP[0.01] and 0.1828 for MAiP, respectively. Tables 14 and 15 show the BM25W obtained the highest scores for the INEX-Wiki09 on 2010 topics for the MATCH EXTENDED mode on the Leaf-Node index and the MATCH ANY mode on the Selected Weight index, with 0.3909 for iP[0.01] and 0.0750 for MAiP, respectively.

We performed a comparative study of the Sphinx search modes based on the MEXIR system. Based on the INEX evaluations on iP[0.01], the results showed that the MATCH EXTENDED mode performed better than all of the other methods for the BM25 function on the Leaf-Node index. Meanwhile, based on the INEX evaluations on MAiP, the MATCH ANY mode performed better than all of the other modes for term frequencies on the Selected Weight index.

Table 17 The Effectiveness of the INEX-2010 Focused Task.

RUN ID	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
BM25	**0.3051**	0.2090	0.1328	0.0512
BM25F	**0.3132**	0.2148	0.1655	0.0540
BM25W	**0.3317**	0.2315	0.1889	0.0550
BM25F-SS	**0.3732**	0.2710	0.2088	0.0730
BM25W-SS	**0.3909**	0.2724	0.2089	0.0750

In the next step, we compared the BM25W with the baseline BM25 and

the BM25F scoring functions. The BM25F needed additional parameter tuning for the W_f, so we set the tuned weights for the BM25F to be W_{title} = 4.0 and W_{body} = 1.2, following (Itakura and Clarke, 2010b). Here, we report the effectiveness of our ranking method with respect to the INEX collections. Table 16 shows the results obtained from the baseline BM25, the BM25F over two fields: the "title" and "body" fields, and the BM25W that used mixed content elements ranking functions on the INEX-Wiki06, while Table 17 shows the results for the INEX-Wiki09.

In order to deepen the analysis of the BM25W scoring function, we also ran experiments to study the impact of mixed content elements on performance. Due to the length of mixed context elements, the double scoring exhibited an improvement of effectiveness as follows: we achieved good iP[0.01] and MAiP, for the baseline BM25 measured in terms of iP[0.01] and MAiP; the improvements were 26.59% and 23.85%, respectively.

Table 18 The Significance (P) is computed with a two-tailed t-test at iP[0.01].

RUN ID	INEX-Wiki06	INEX-Wiki09
BM25F-SS	0.5678	0.3732
BM25W-SS	0.6499	0.3909
P(t-test)	**0.04**	**0.14**

Table 19 A comparison with GPX in the INEX-2008 Focused Task.

RUN ID	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
GPX	**0.6344**	0.5693	0.5178	0.2587
BM25W-SS	**0.6499**	0.5724	0.5438	0.1828

The double scoring performed well with the INEX-Wiki06, which had a relatively larger size of mixed content elements for each document (see Table 11). It can be seen that BM25W obtained the best performance, although the improvement over both the baseline BM25 and the BM25F was significant for most of the considered metrics. The Significance (P) was computed with a two-tailed t-test; the BM25W improved by 0.04% over the BM25F at iP[0.01] on the INEX-Wiki06, and by 0.14% over the BM25F at iP[0.01] on the INEX-Wiki09 as shown in Table 18.

In this analysis, we take the results that were obtained from BM25W over mixed content elements and compare them with the results from BM25F over two fields, i.e. “title” and “body”. It is shown again that BM25W worked well with the mixed content elements of the document-centric XML documents. We can conclude that significant improvement of results of the BM25W can be obtained from mixed content elements. This finding suggests that it is possible to improve the BM25W approach, which is the usual benchmark in the INEX. The main conclusion that can be drawn from the experiments is that the Double scoring algorithm is successful in automatically selecting document fields by using mixed content elements and in assigning the weight to each chosen field at query time.

In addition, applying the score sharing technique to this function, the experiments showed a 7.27% improvement over the BM25W, a 9.29% improvement over the BM25F measured by iP[0.01] on the INEX-Wikip06, and a 17.84% improvement over the BM25W, a 19.15% improvement over the BM25F measured by iP[0.01] on the INEX-Wikip09. Furthermore, in comparing the effectiveness to the GPX system, the BM25W-SS showed a 2.44% improvement at iP[0.01] on the INEX-Wiki06 as shown in Table 19.

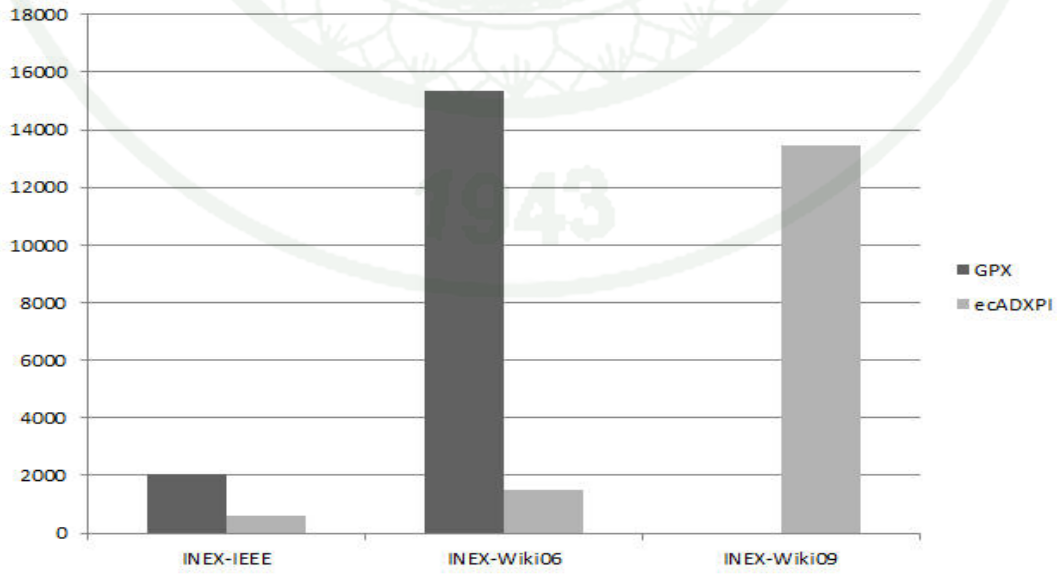


Figure 29 A comparison with GPX in the data size (MB)

3. Storage and Efficiency Results and Discussion

In the following section, we discuss the results of the compression technique focusing on the storage and response time. First, we consider the complexity and regularity of the XML documents. Based on the entropy in function (49) (Shannon, 1948) of the individual elements, the complexity of all the XML documents can be estimated for each collection of the INEX as shown in Table 20.

Table 20 The Structural Entropy in the INEX Collections.

Collections	Tag + Position	Tag Only
INEX-IEEE	4.6644	3.2069
INEX-Wiki06	3.2780	2.3385
INEX-Wiki09	8.2050	7.5709

Table 21 Distribution of Elements in INEX Collections.

Collections	Tag Mapping	Element	N_{freq}
INEX-IEEE	0	article	1494676
	1	bdy	1370545
	2	sec	1364184
	3	p	759925
	4	ss1	691972
	5	ip1	245506
INEX-Wiki06	0	article	6360427
	1	body	5704185
	2	section	5066779
	3	p	2559512
	4	title	1545969
	5	name	656295
INEX-Wiki09	0	p	25759610
	1	sec	25580700
	2	article	16777220
	3	bdy	16777220
	4	template	10022210
	5	parameters	9219120

We present the compression technique with the distribution of elements over element-names and the numbers of their positions. In this case, the most frequent are mapped to a small numbers in the compression method as shown in Table 21.

As a result, it is clear that less storage is required if less data is stored in the index. Table 22 and Figure 29 show that the savings are significant: the use of compression technique reduces the data size by 90.19% compared to the GPX system (Geva, 2007). Table 23 shows the data sizes with and without compression. Data compression also reduces the length of the score sharing function processing time by 37.12% when compared to before compression, as shown in Table 24 and Figure 30.

Table 22 A comparison of the data size to GPX.

Collections	**GPX**	**ecADXPI**	**%**
INEX-IEEE	2,048	597	**70.84**
INEX-Wiki06	15,360	1,506	**90.19**
INEX-Wiki09	-	13,432	**-**

Table 23 A comparison of data size with and without compression.

Collections	**ADXPI**	**ecADXPI**	**%**
INEX-IEEE	629	597	**5.09**
INEX-Wiki06	1,910	1,506	**21.15**
INEX-Wiki09	16,199	13,432	**17.08**

To start with the compression technique, we emphasize the system performance in the experimental results of our approach for retrieving on large collections, to improve the efficiency of XML Retrieval. The performance of the compression model was evaluated and we can summarize our findings as follows:

1. The compression technique showed a 70.84% improvement over GPX on the INEX-IEEE, and a 90.19% improvement over GPX on the INEX-Wiki06 (Geva, 2007). However, no experiment was performed with the GPX system on the INEX-Wiki09, so we cannot report for this collection.

2. The compression technique showed a 5.09% improvement over ADXPI on the INEX-IEEE, a 21.15% improvement over ADXPI on the INEX-Wiki06, and a 17.08% improvement over ADXPI on the INEX-Wiki09.

3. Another conclusion is that compression also reduced the length of the Score Sharing function processing time by 3.45%, 12.82%, 6.73%, 5.93%, 15.35% and 37.12% with respect to the respective numbers of leaf nodes: 2500, 5000, 7500, 10000, 50000, and 100000, when compared the time to before using the compression technique.

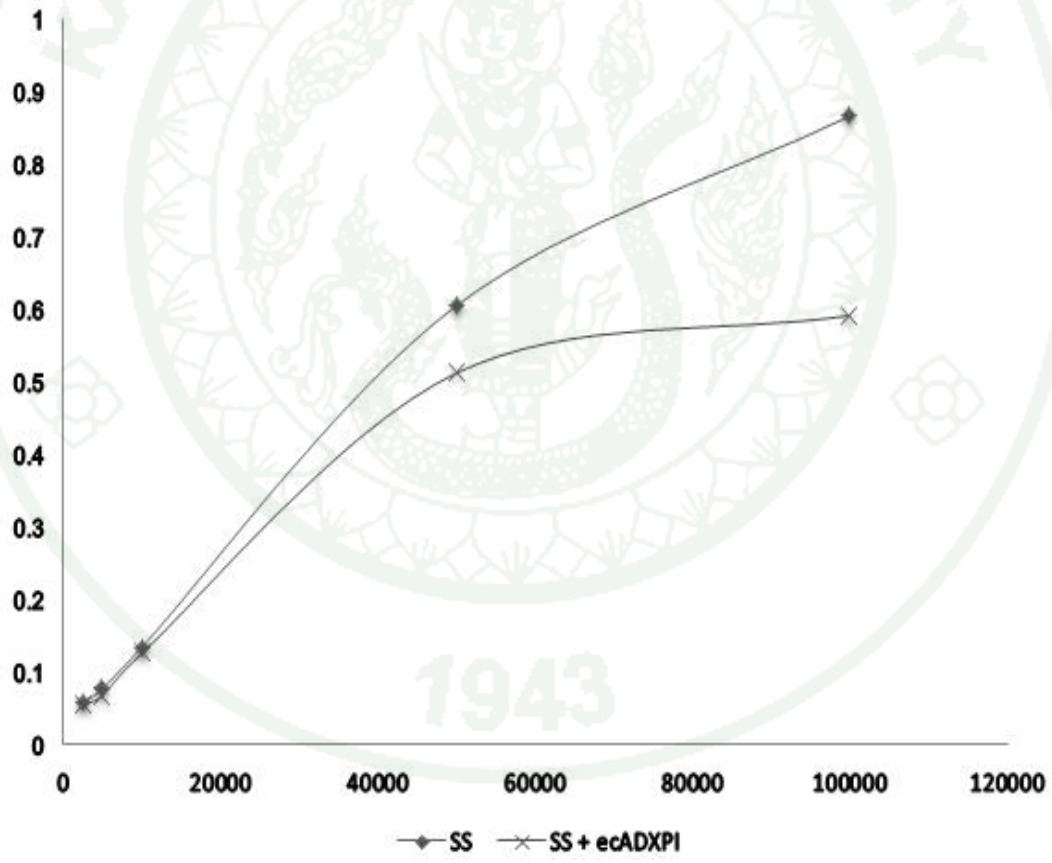


Figure 30 A comparison of Score Sharing Processing Time (ms.) with and without compression at β=0.10.

In addition, in terms of processing time, the MEXIR system required an average of one second per topic in the INEX-IEEE and four seconds per topic in the INEX-Wiki06. This was better than the GPX system, a reduction of 44.44% with respect to the environment resources as shown in Table 25 compared to the GPX system.

Table 24 A comparison of Score Sharing Processing Time at β=0.10 (ms).

N	ADXPI	ecADXPI	Decompress	%
2500	0.058	0.041	0.015	**3.45**
5000	0.078	0.043	0.025	**12.82**
7500	0.104	0.067	0.030	**6.73**
10000	0.135	0.092	0.035	**5.93**
50000	0.606	0.147	0.366	**15.35**
100000	0.867	0.216	0.376	**31.72**

Table 25 A comparison of environment resources to GPX.

Resource	GPX	MEXIR
CPU	3 GHz	Dual-Core 1.87 GHz
RAM	2 GB	1 GB

Table 26 A comparison with GPX in terms of Response Time (sec.).

Collections	GPX	MEXIR	%
INEX-IEEE	-	1	**-**
INEX-Wiki06	7.2	4.0	**44.44**

4. MEXIR at the INEX Results and Discussion

Extensive experiments on various real-world data collections, as well as the INEX official benchmark settings for both text and structured data demonstrate the increased efficiency, effectiveness, and scalability of our approach. In this section, we report our participation in the INEX. We participated in three tracks: the Ad hoc track, the Data Centric track and the Web Service Discovery track as follows:

4.1 Ad hoc Track

On the Ad hoc track, we present the results used to evaluate the MEXIR retrieval system. The MEXIR search engine retrieves XML elements based on the leaf node indexed with respect to the significant words, includes the double scoring and score sharing function, and combines the relevance score of a leaf element with the score of its parent element. Finally, the element with the higher relevance score will be chosen as the retrieval set because it has no nested elements. In principle, any portion of an XML document can be retrieved, although some portions are more likely to be relevant to a user's query. Table 27 shows a comparison of BM25W to the top ten of INEX participants. Our run **BM25W-SS** for the Ad Hoc Task ranked seventh best, scoring 0.6499 measured with iP[0.01] with respect to the INEX (Kamps *et al.*, 2009).

Table 27 A comparison of the top 10 participants in the INEX-2008 Ad Hoc Track Focused Task.

Authors	**iP[0.01]**	**iP[0.05]**	**iP[0.10]**	**MAiP**
(Itakura and Clarke, 2009)	0.6873	0.5700	0.4879	0.2071
(Theobald *et al.*, 2009)	0.6795	0.5807	0.5265	0.2967
(Verbyst and Mulhem, 2009)	0.6665	0.5210	0.4216	0.1441
(Ibekwe-SanJuan and SanJuan, 2009)	0.6664	0.6139	0.5540	0.3065
(Géry *et al.*, 2009)	0.6640	0.5800	0.4986	0.2342
(Lehtonen and Doucet, 2009)	0.6619	0.5532	0.5028	0.2251
BM25W-SS	**0.6499**	**0.5724**	**0.5438**	**0.1828**
(Larson, 2009)	0.6395	0.4906	0.4026	0.1392
(Kamps *et al.*, 2009)	0.6346	0.5490	0.5222	0.2647
(Kamps *et al.*, 2009)	0.6344	0.5693	0.5178	0.2587
(Pal *et al.*, 2009)	0.6337	0.5537	0.5100	0.2847

4.2 Data Centric Track

On the Data Centric (DC) track, we used the values $k1 = 1.80$ and $b = 0.40$ to evaluate the sensitivity of the element length of BM25 on both indices. The total number of leaf nodes was 2500 and the β parameter was set to 1.50, which was used to compute the sharing score. We found that all of the content had quite a few data. Then we turned to the document level as the better answer for the user. The MEXIR search engine retrieves XML elements based on the leaf node index with respect to significant

words and the score sharing functions, and then combines the relevance score of the element in the document score. Thus, the document with the highest relevance score is chosen as the retrieval set.

Table 28 Best performing runs based on MAP over the information topics.

Authors	MAP	1/Rank	P@10	P@20
(Wichaiwong and Jaruskulchai, 2012c)	**0.3564**	**0.8000**	**0.5000**	**0.4200**
(Theobald *et al.*, 2012)	0.3449	0.7067	0.5000	0.4700
(Ayala *et al.*, 2012)	0.3219	1.0000	0.5600	0.4300
(Wang *et al.*, 2012)	0.3189	0.6500	0.4200	0.4500
(Schuth and Marx, 2012)	0.3079	0.6750	0.3800	0.3100
(Ramírez, 2012)	0.2576	0.6346	0.4600	0.4400
(Theobald *et al.*, 2012)	0.2118	0.5015	0.4400	0.4200
(Theobald *et al.*, 2012)	0.0900	0.3890	0.2600	0.1800
(Laitang *et al.*, 2012)	0.0366	0.3022	0.2200	0.1100

The INEX has provided an excellent test corpus on XML information retrieval and queries. The Data Centric Track (Trotman and Wang, 2011) investigates retrieval over a strongly structured collection of documents based on the IMDB collection. This task has a total of three topics, one for each of the following categories:- *Known-item*: For these topics, the relevant answer is a single document; *List*: Topics that ask for a list of objects; and *Informational*: Topics that ask for information about any topic/movie/person contained in the collection.

Table 29 Best performing runs based on 1/Rank over the known-item topics.

Authors	MAP	1/Rank	P@10	P@20
(Schuth and Marx, 2012)	0.8112	0.9167	0.3167	0.2417
(Wang *et al.*, 2012)	0.7264	0.9167	0.3167	0.2417
(Theobald *et al.*, 2012)	0.2916	0.7222	0.2333	0.1833
(Ramírez, 2012)	0.3752	0.7104	0.2500	0.2083
(Wichaiwong and Jaruskulchai, 2012c)	**0.4745**	**0.6667**	**0.0833**	**0.0417**
(Theobald *et al.*, 2012)	0.5492	0.6389	0.3167	0.2417
(Theobald *et al.*, 2012)	0.3100	0.5730	0.2667	0.1750
(Ayala *et al.*, 2012)	0.2500	0.3333	0.0333	0.0167
(Laitang *et al.*, 2012)	0.0221	0.0487	0.0167	0.0333

We analyzed the effectiveness of the runs for each of the three topic types

with respect to the INEX (Theobald *et al.*, 2012) and the results are presented in Tables 28, 29, 30. The overall results are satisfactory if we compare them with those obtained by participants in the INEX contests. In comparing the effectiveness for the informational topics, our run ranked first, scoring 0.3564, measured with MAP, it ranked fifth scoring 0.6667 measured with 1/Rank for the known-item topics, and in the results of the list topics, our run ranked first, scoring 0.4251, measured with MAP.

In this analysis, we take the results that were obtained from the INEX report (Theobald *et al.*, 2012). It is shown again that our system works well with the *List* and *Informational* topics of the document-centric XML documents measured with the MAP metric. Unfortunately, on the *Known-item* topics, the relevant answer is a single document; in this area, the performance was not satisfactory and so further investigation is required.

Table 30 Best performing runs based on MAP over the list topics.

Authors	**MAP**	**1/Rank**	**P@10**	**P@20**
(Wichaiwong and Jaruskulchai, 2012c)	**0.4251**	**0.7778**	**0.4778**	**0.3833**
(Schuth and Marx, 2012)	0.3454	0.6674	0.4222	0.3500
(Theobald *et al.*, 2012)	0.3332	0.5432	0.3889	0.3667
(Wang *et al.*, 2012)	0.3264	0.6488	0.4111	0.3333
(Theobald *et al.*, 2012)	0.2578	0.4926	0.3000	0.3333
(Ramírez, 2012)	0.2242	0.5756	0.3556	0.3278
(Laitang *et al.*, 2012)	0.1532	0.2542	0.2333	0.2111
(Theobald *et al.*, 2012)	0.0847	0.5027	0.1889	0.1611
(Ayala *et al.*, 2012)	0.0798	0.3902	0.2889	0.2500

4.3 Web Service Discovery Track

On the Web Service Discovery track, standard tokens cannot be utilized directly due to various reasons such as sub-language patterns and programming conventions. Therefore, these tokens need to be converted to natural languages before being indexed using IR models.

The Capitalization Styles in (Wichaiwong *et al.*, 2008) utilized three types of capitalisation styles;*UpperCamelCase* (Pascal), wherein the first letter is the

identifier and the first letters of each subsequent concatenated word are capitalized, such as the “BackColor”, “DataSet”; *lowerCamelCase* (Camel), which means that the first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalised, such as “backColor”, “dataSet”; and *UpperCase* (Upper), in which all of the letters in the identifier are capitalised, such as “BACKCOLOR”, “DATASET”.

Table 31 The details of the Web Service Experiments.

RUN ID	Sphinx Mode	β of Score Sharing
Kas-I138BM25ESS010	EXTENDED	0.10
Kas-I138ANYBM25SS015	ANY	0.15
Kas-I138ANYSS025	ANY	0.25

Table 32 A comparison of performance runs based on MAP of the INEX-2010 Web Service Track.

Authors	RUN ID	MAP
(Wichaiwong and Jaruskulchai, 2011e)	**Kas-I138BM25ESS010**	0.3469
(Thom and Wu, 2011)	RMIT10WS	0.2125
(Thom and Wu, 2011)	HPI2010	0.2946
(Wichaiwong and Jaruskulchai, 2011e)	**Kas-I138ANYBM25SS015**	0.2798
(Wichaiwong and Jaruskulchai, 2011e)	**Kas-I138ANYSS025**	0.2798
(Hou *et al.*, 2011)	QUT-BM25WordComp	0.2348
(Hou *et al.*, 2011)	QUT-BM25Word	0.2233
(Hou *et al.*, 2011)	QUT-BM25WordCompNet	0.2042
(Somodevilla *et al.*, 2011)	BUAPFCCWSD01	0.1451
(Hou *et al.*, 2011)	QUT-Wikipedia	0.1268
(Hou *et al.*, 2011)	QUT-WikipediaComp	0.1095
(Hou *et al.*, 2011)	QUT-WikipediaCompNet	0.0937

Almost all of the web service methods have already utilized the form of both “UpperCamelCase” and“lowerCamelCase”, such as naming conventions in several programming languages. For instance, applying the Capitalisation function (Wichaiwong *et al.*, 2008) for the “msdGetXsdBase64BinData” and then applying the above function gives the result “msd”, “Get”, “Xsd”, “Base64”, “Bin” and “Data” and another example of the document “0351” as shown in Figure 31.

We applied this function to solve the tokenisation issue for both elements

and attributes, and then we present an extension to the MEXIR framework that supports the extraction attributes of the WSDL documents. Afterwards, we submitted three runs to this track, the details of which are shown in Table 31. As a result, our runs showed that “Kas-I138BM25ESS010” ranked first with MAP at 0.3469, “Kas-I138ANYBM25SS015” ranked fourth with MAP at 0.2798 and our other run “Kas-I138ANYSS025” ranked fifth with MAP at 0.2798, as shown in Table 32.

```
<?xml version="1.0"?>
<definitions name="InteropDIMETest"
    targetNamespace="http://soapinterop.org/attachments/wsdl"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    xmlns:tns="http://soapinterop.org/attachments/wsdl"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">

    <import location="http://www.pocketsoap.com/interop/dime-rpc.wsdl" namespace=
"http://soapinterop.org/attachments/wsdl"/>

    <service name="DimeInteropRPCService">

        <port name="DimeInteropRPCTestPort" binding="tns:AttachmentsBinding">
            <soap:address location="http://www.whitemesa.net/interop/r4/dimerpc"/>
        </port>

    </service>

</definitions>
```

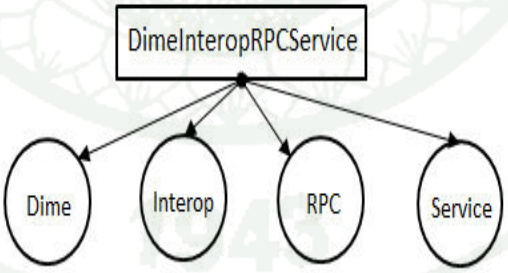


Figure 31 Illustrations of the capitalization processing

CONCLUSION AND RECOMMENDATION

Conclusion

In this thesis, we have presented various aspects of content-only element retrieval, ranking from indexing and the implementation of an XML retrieval system. Our expectation is that novel methods specifically designed for XML retrieval will improve both efficiency and effectiveness of search systems. The most surprising results of this thesis with respect to the objectives are as follows:

We begin by addressing the XML-IR issue of discarding some small units of information and nested elements by discussing the results concerned with the element granularity returned to users. An element is considered the best result when it has the coverage, and the highest density of terms contained in the query. The relative effectiveness of the results obtained from the Score Sharing algorithm result may be considered in terms of how well they work with document-centric XML documents. This method is based on relevance scores calculated for each leaf node and then shared with their parents by using the distance between nodes. A significant improvement of results of the classical IR methods can be obtained by the Score Sharing method. In line with this observation, these experiments were aimed at having the Scoring Sharing function automatically return the appropriate level of the XML document. However, XML elements also show variations in the length of their context and those of their children. As a result, elements containing a lot of children are ranked above those with only a few child nodes, even though they are not as relevant. A way to handle this is by normalizing the length of the element and filtering out the noise elements that are not likely to reflect its actual content. Thus, we need an indicator of the element score together with the length of the element, and this has a great deal of potential in XML retrieval systems. This requires further investigation.

A second aim was to address the research issues of the selected type approach, in which choosing fields and assigning weights are performed manually. We propose two automatic methods which make it possible to tune the weight of fields in document-centric XML documents by using mixed content elements and assigning weights to

the selected fields. These approaches achieve an improvement of up to 15% over the BM25F based on the INEX evaluations. Here again, the Double Scoring function works well with the mixed content elements of the document-centric XML documents. A significant improvement of results of the Double Scoring algorithm can be obtained from the mixed content elements. The main conclusion that can be drawn from the experiments is that the Double Scoring function is successful in automatically selecting document fields by using mixed content elements and assigning the weight for each chosen field at query time. However, a bias smoothing parameter in relation to rewarding and penalizing each index has not been used as a normalization function incorporated into the Double Scoring algorithm in the ranking function. This also requires further investigation.

Lastly, even though the XML structure has benefits, it can be space and time consuming as well, and this has a bearing on the efficiency of the search system. The main disadvantage of using XML documents is their large size caused by the highly structured nature of such documents, which often use long tags and attribute names. We propose a new XML compression algorithm that accommodates ADXPI indexing and the Score Sharing algorithm. Essentially, these steps reduce the size of the data by 90.19%, as compared with the GPX search system. Further, they reduce the score sharing processing time by 37.12% as compared with the processing time before compression. Since the size of the data is reduced, processing the compressed XML data and paths may be even be faster than processing the data in its original form. In addition, this technique is able to support content-and-structure queries that allow retrieval of the path in the compressed data directly. It would be particularly advantageous to apply the bit string to the compression technique by using bit operators. This could result in an overall improvement in system efficiency.

Recommendation

There are many issues of XML retrieval addressed in this thesis, but its main focus is to study and develop support for content-only queries. The next logical step would be an extension to the MEXIR system to support ontology and make a step towards content-and-structure queries and then to full-text queries that would support the standard XML query languages such as NEXI and XQuery.

LITERATURE CITED

Amer-Yahia, S. and M. Lalmas. 2006. XML search: languages, INEX and scoring. **SIGMOD Rec**. 35(4):16–23.

Anh, V. N. and A. Moffat. 2002. Compression and an IR Approach to XML Retrieval, pp. 99–104. *In* N. Fuhr, N. Gövert, G. Kazai and M. Lalmas, eds., **Proceedings of the First Workshop of the INitiative for the Evaluation of XML Retrieval (INEX)**. Dagstuhl, Germany.

Arvola, P., M. Junkkari and J. Kekäläinen. 2005. Generalized contextualization method for XML information retrieval, pp. 20–27. *In* **Proceedings of the 14th ACM international conference on Information and knowledge management**. CIKM '05. ACM, New York, NY, USA.

Arvola, P., J. Kekäläinen and M. Junkkari. 2011. Contextualization models for XML retrieval. **Information Processing and Management**. 47(5):762–776.

Ayala, D. V., D. Pinto, S. L. Silverio, E. Castillo and M. T. Vidal. 2012. BUAP: A Recursive Approach to the Data-Centric Track of INEX 2011, pp. 161–166. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Baeza-Yates, R. A. and B. Ribeiro-Neto. 1999. **Modern Information Retrieval**. Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Booth, D., H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris and D. Orchard. 2004. **Web Services Architecture**. Available Source: http://www.w3.org/TR/ws-arch/, May 05, 2012.

Booth, D. and C. K. Liu. 2007. **Web Services Description Language (WSDL) Version 2.0**. Available Source: http://www.w3.org/TR/2007/, August 23, 2012.

Bray, T., J. Paoli, C. Sperberg-McQueen, E. Maler, F. Yergeau and J. Cowan. 2008. **Extensible Markup Language (XML) 1.1 (Fifth Edition)**. Available Source: http://www.w3.org/TR/xml/, August 23, 2012.

Broglio, J., J. P. Callan, W. B. Croft and D. W. Nachbar. 1994a. Document Retrieval and Routing Using the INQUERY System, pp. 29–38. *In* **In Proceeding of Third Text Retrieval Conference (TREC-3**.

Broglio, J., J. P. Callan, W. B. Croft and D. W. Nachbar. 1994b. Document Retrieval and Routing Using the INQUERY System, pp. 29–38. *In* **In Proceeding of Third Text Retrieval Conference (TREC-3)**.

Broschart, A. and R. Schenkel. 2008. Proximity-aware scoring for XML retrieval, pp. 845–846. *In* **Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval**. SIGIR '08. New York, NY, USA.

Broschart, A. and R. Schenkel. 2010. Index tuning for efficient proximity-enhanced query processing, pp. 213–217. *In* **Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval**. INEX'09. Springer-Verlag, Berlin, Heidelberg.

Buffoni, D., N. Usunier and P. Gallinari. 2010. LIP6 at INEX'09: OWPC for ad hoc track, pp. 59–69. *In* **Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval**. INEX'09. Springer-Verlag, Berlin, Heidelberg.

Craswell, N., H. Zaragoza and S. Robertson. 2005. Microsoft Cambridge at TREC 14: Enterprise Track. *In* V. E. M. and B. L. P., eds., **TREC**, vol. Special Publication 500-266. National Institute of Standards and Technology (NIST).

Crouch, C. J.. 2006. Dynamic element retrieval in a structured environment. **ACM Transactions on Information Systems**. 24(4):437–454.

Denoyer, L. and P. Gallinari. 2006. The Wikipedia XML corpus, pp. 64–69. *In* **SIGIR Forum**.

Dongwook, S., J. Hyuncheol and J. Honglan. 1998. BUS: an effective indexing and retrieval scheme in structured documents, pp. 235–243. *In* W. Ian, A. Rob and S. F. M., eds., **Proceedings of the third ACM conference on Digital libraries**. ACM Press, Pittsburgh, Pennsylvania, USA.

Fuhr, N. and N. Gövert. 2002. Index compression vs. retrieval time of inverted files for XML documents, pp. 662–664. *In* **Proceedings of the eleventh international conference on Information and knowledge management**. CIKM '02. ACM, New York, NY, USA.

Fuhr, N., J. Kamps, M. Lalmas, S. Malik and A. Trotman. 2008. Overview of the INEX 2007 Ad Hoc Track. *In* N. Fuhr, J. Kamps, M. Lalmas and A. Trotman, eds., **Focused Access to XML Documents: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 1–23. Springer-Verlag, Berlin, Heidelberg.

Gailly, J. L. and M. Adler. 2010. **gzip: The compressor data**. Available Source: http://www.gzip.org, May 05, 2012.

Géry, M. and C. Largeron. 2010. UJM at INEX 2009 ad hoc track, pp. 88–94. *In* **Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval**. INEX'09. Springer-Verlag, Berlin, Heidelberg.

Géry, M., C. Largeron and F. Thollard. 2009. UJM at INEX 2008: Pre-impacting of Tags Weights. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 46–53. Berlin, Heidelberg.

Geva, S.. 2005. GPX - Gardens Point XML Information Retrieval at INEX 2004, pp. 211–223. *In* **Advances in XML Information Retrieval: Third International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Geva, S.. 2006. GPX - Gardens Point XML Information Retrieval at INEX 2005, pp. 240–253. *In* **The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Geva, S.. 2007. GPX - Gardens Point XML Information Retrieval at INEX 2006, pp. 137–150. *In* **Comparative Evaluation of XML information Retrieval Systems: The 5th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Geva, S., J. Kamps, M. Lehtonen, R. Schenkel, J. A. Thom and A. Trotman. 2010. Overview of the INEX 2009 Ad Hoc Track, pp. 4–25. *In* S. Geva, J. Kamps and A. Trotman, eds., **Focused Retrieval and Evaluation: 8th International Workshop of the Initiative for the Evaluation of XML Retrieval**. Springer-Verlag, Berlin, Heidelberg.

Geva, S., J. Kamps and R. Schenkel. 2011. **INitiative for the Evaluation of XML Retrieval (INEX)**. Available Source: https://inex.mmci.uni-saarland.de/, May 05, 2012.

Hou, J., J. Zhang, R. Nayak and A. Bose. 2011. Semantics-based web service discovery using information retrieval techniques, pp. 336–346. *In* **Comparative Evaluation of Focused Retrieval : The 9th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Ibekwe-SanJuan, F. and E. SanJuan. 2009. Use of Multiword Terms and Query Expansion for Interactive Information Retrieval. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 54–64. Berlin, Heidelberg.

Itakura, K. Y. and C. L. Clarke. 2009. University of Waterloo at INEX 2008: Adhoc, Book, and Link-the-Wiki Tracks. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 132–139. Berlin, Heidelberg.

Itakura, K. Y. and C. L. A. Clarke. 2010a. A framework for BM25F-based XML retrieval, pp. 843–844. *In* **Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval**. Geneva, Switzerland.

Itakura, K. Y. and C. L. A. Clarke. 2010b. University of waterloo at INEX 2009: ad hoc, book, entity ranking, and link-the-wiki tracks, pp. 331–341. *In* **Proceedings of the Focused retrieval and evaluation, and 8th international conference on Initiative for the evaluation of XML retrieval**. INEX'09. Springer-Verlag, Berlin, Heidelberg.

Järvelin, K. and J. Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. **ACM Transactions on Information Systems**. 20(4):422–446.

Kamps, J., de Maarten De Rijke and B. Sigurbjörnsson. 2005a. The Importance of Length Normalization for XML Retrieval. **Information Retrieval**. 8:631–654.

Kamps, J., M. de Rijke and B. Sigurbjörnsson. 2004. Length normalization in XML retrieval, pp. 80–87. *In* **Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval**. SIGIR '04. ACM, New York, NY, USA.

Kamps, J., S. Geva, A. Trotman, A. Woodley and M. Koolen. 2009. Overview of the INEX 2008 Ad Hoc Track. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 1–28. Springer-Verlag, Berlin, Heidelberg.

Kamps, J., M. Koolen and M. Lalmas. 2007a. Where to start reading a textual XML document?, pp. 723–724. *In* **Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval**. SIGIR '07. ACM, New York, NY, USA.

Kamps, J., M. Marx, M. de Rijke and B. Sigurbjörnsson. 2005b. Structured queries in XML retrieval, pp. 4–11. *In* **Proceedings of the 14th ACM international conference on Information and knowledge management**. CIKM '05. ACM, New York, NY, USA.

Kamps, J., J. Pehcevski, G. Kazai, M. Lalmas and S. Robertson. 2007b. INEX 2007 evaluation measures, pp. 24–33. *In* **Focused Access to XML Documents: The 6th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Kazai, G. and M. Lalmas. 2006. INEX 2005 Evaluation Measures, pp. 16–29. *In* **Advances in XML Information Retrieval and Evaluation, 4th International Workshop of the Initiative for the Evaluation of XML Retrieval, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Khanna, S.. 2005. **Design and implementation of a flexible retrieval system**. Master's thesis. Department of Computer Science, University of Minnesota Duluth.

Koolen, M., R. Kaptein and J. Kamps. 2009. University of Amsterdam at INEX 2009: Ad hoc, Book, and Entity Ranking Tracks, pp. 260–272. *In* S. Geva, J. Kamps and A. Trotman, eds., **INEX 2009 Workshop Pre-proceedings**.

Laitang, C., K. Pinel-Sauvagnat and M. Boughanem. 2012. Edit Distance for XML Information Retrieval: Some Experiments on the Datacentric Track of INEX 2011, pp. 138–145. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Lalmas, M.. 2009. Term Statistics for Structured Text Retrieval. *In* **Encyclopedia of Database Systems**. pp. 3036–3037.

Larson, R. R.. 2009. Adhoc and Book XML Retrieval with Cheshire. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 152–163. Berlin, Heidelberg.

Lehtonen, M. and A. Doucet. 2009. Enhancing Keyword Search with a Keyphrase Index. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 65–70. Berlin, Heidelberg.

Liefke, H. and D. Suciu. 2000. XMill: An Efficient Compressor for XML Data, pp. 153–164. *In* **Proceeding of the 2000 ACM SIGMOD International Conference on Management of Data**.

Lu, W., S. Robertson and A. MacFarlane. 2005. Field-Weighted XML Retrieval Based on BM25, pp. 161–171. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Mairieng, K. and C. Pluempitiwiriyawej. 2003. XPACK: A Grammar-based XML Document Compression, pp. 161–171. *In* **Proceeding of NCSEC2003 the7th National Computer Science and Engineering Conference**. Chonburi, Thailand.

Mani, A. and A. Nagarajan. 2002. **Understanding quality of service for Web services**. Available Source: http://www.ibm.com/webservices/index.html, May 05, 2012.

Manning, C. D., P. Raghavan and H. Schütze. 2008. **Introduction to Information Retrieval**. Cambridge University Press.

Mass, Y. and M. Mandelbrod. 2006. Using the INEX Environment as a Test Bed for Various User Models for XML Retrieval, pp. 187–195. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Min, J. K., M. J. Park and C. W. Chung. 2003. XPRESS: A Queriable Compression for XML Data, pp. 122–133. *In* **Proceeding of the 2003 ACM SIGMOD International Conference on Management of Data**.

Najork, M. A., H. Zaragoza and M. J. Taylor. 2007. Hits on the web: how does it compare?, pp. 471–478. *In* **Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval**. SIGIR '07. ACM, New York, NY, USA.

Ogilvie, P. and J. Callan. 2005. Hierarchical Language Models for XML Component Retrieval, pp. 118–125. *In* **Advances in XML Information Retrieval: The Third International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Ogilvie, P. and J. Callan. 2006. Parameter Estimation for a Simple Hierarchical Generative Model for XML Retrieval, pp. 211–224. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Pal, S., M. Mitra, D. Ganguly, S. Maiti, A. Bandyopadhyay, A. Sen and S. Mitra. 2009. Indian Statistical Institute at INEX 2008 Adhoc Track. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 79–86. Berlin, Heidelberg.

Pehcevski, J. and J. A. Thom. 2005. Hixeval: Highlighting XML retrieval evaluation, pp. 43–57. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Pehcevski, J. and J. A. Thom. 2006. HiXEval: highlighting XML retrieval evaluation, pp. 43–57. *In* **Proceedings of the 4th international conference on Initiative for the Evaluation of XML Retrieval**. Springer-Verlag, Berlin, Heidelberg.

Pérez-Agüera, J. R., J. Arroyo, J. Greenberg, J. P. Iglesias and V. Fresno. 2010. Using BM25F for semantic search, pp. 21–28. *In* **Proceedings of the 3rd International Semantic Search Workshop**. SEMSEARCH '10. ACM, New York, NY, USA.

Pharo, N.. 2008. The effect of granularity and order in XML element retrieval. **Information Processing and Management**. 44(5):1732–1740.

Ramírez, G.. 2012. UPF at INEX 2011: Books and Social Search Track and Data-Centric Track, pp. 146–154. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Rzadkiewicz, C.. 2011. **How to Read a College Textbook Fast yet Comprehend Key Points**. Available Source: http://suite101.com/article/how-to-read-a-college-textbook-fast-yet-comprehend-key-points-a358334/, October 14, 2012.

Sakr, S.. 2009. XML compression techniques: A survey and comparison. **Journal of Computer and System Sciences**. 75(5):303–322.

Salton, G.. 1965. The evaluation of automatic retrieval procedures-selected test results using the SMART system. **American Documentation**. 16:209–222.

Schenkel, R., F. M. Suchanek and G. Kasneci. 2007. YAWN: A Semantically Annotated Wikipedia XML Corpus., pp. 277–291. *In* **Proceedings of Datenbanksysteme in Business, Technologie and Web (BTW'07)**.

Schuth, A. and M. Marx. 2012. University of Amsterdam Data Centric Ad Hoc and Faceted Search Runs, pp. 155–160. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Shannon, C. E.. 1948. A mathematical theory of Communication. **The Bell system technical journal**. 27:379–423.

Sigurbjörnsson, B. and J. Kamps. 2005. The effect of structured queries and selective indexing on XML retrieval, pp. 104–118. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Sigurbjörnsson, B., J. Kamps and M. de Rijke. 2004. An Element-based Approach to XML Retrieval, pp. 19–26. *In* N. Fuhr, S. Malik and M. Lalmas, eds., **INEX 2003 Workshop Proceedings**.

Sigurbjörnsson, B., J. Kamps and M. de Rijke. 2004. Processing content-oriented XPath queries, pp. 371–380. *In* **Proceedings of the thirteenth ACM international conference on Information and knowledge management**. CIKM '04. ACM, New York, NY, USA.

Singhal, A., C. Buckley and M. Mitra. 1996. Pivoted document length normalization, pp. 21–29. *In* **Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval**. SIGIR '96. New York, NY, USA.

Somodevilla, M. J., B. Beltrán, D. Pinto, D. Vilariño and J. C. Aaron. 2011. The BUAP participation at the web service discovery track of INEX 2010, pp. 347–350. *In* **Comparative Evaluation of Focused Retrieval : The 9th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Stephen, R. and Z. Hugo. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. **Foundations and Trends in Information Retrieval**. 3(4):333–389.

Stephen, R., Z. Hugo and T. Michael. 2004. Simple BM25 extension to multiple weighted fields, pp. 42–49. *In* D. Grossman, L. Gravano, C. Zhai, O. Herzog and D. A. Evans, eds., **CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management**. ACM Press, New York, NY, USA.

Stephen, R., S. Walker, S. Jones, M. Hancock-Beaulieu and M. Gatford. 1994. Okapi at TREC-3, pp. 109–126. *In* **Overview of the Third Text REtrieval Conference**. National Institute of Standards and Technology (NIST).

Theobald, A. and G. Weikum. 2002. The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking, pp. 477–495. *In* **Proceedings of the 8th International Conference on Extending Database Technology: Advances in Database Technology**. Springer-Verlag, London, UK, UK.

Theobald, M.. 2006. **Efficient and Versatile Top-K Query Processing for Text, Structured, and Semistructured Data.** Ph.D. thesis. Universität des Saarlandes.

Theobald, M., M. AbuJarour and R. Schenkel. 2009. TopX 2.0 at the INEX 2008 Efficiency Track: A (Very) Fast Object-Store for Top-k-Style XML Full-Text Search. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 224–236. Berlin, Heidelberg.

Theobald, M., H. Bast, D. Majumdar, R. Schenkel and G. Weikum. 2008. TopX: efficient and versatile top-query processing for semistructured data. **International Journal on Very Large Data Bases**. pp. 81–115.

Theobald, M., R. Schenkel and G. Weikum. 2005. TopX and XXL at INEX 2005, pp. 282–295. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Theobald, M., Q. Wang, G. Ramírez, M. M. Marx, M. Theobald and J. Kamps. 2012. Overview of the INEX 2011 Data-Centric Track. *In* **Focused Retrieval of Content and Structure: 10th International Workshop of the Initiative for the Evaluation of XML Retrieval (INEX 2011)**. Springer, Hofgut Imsbach, Theley, Germany.

Thom, J. A. and C. Wu. 2011. Overview of the INEX 2010 web service discovery track, pp. 332–335. *In* **Comparative Evaluation of Focused Retrieval : The 9th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Tolani, P. M. and J. R. Haritsa. 2002. XGRIND: A Query-Friendly XML Compressor., pp. 225–234. *In* **International Conference on Databases Engineering 2002**.

Trotman, A.. 2005. Choosing document structure weights. **Information Processing and Management**. 41:243–264.

Trotman, A. and M. Lalmas. 2005. The Interpretation of CAS, pp. 58–71. *In* **Advances in XML Information Retrieval and Evaluation: The 4th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Trotman, A. and M. Lalmas. 2006. Why structural hints in queries do not help XML-retrieval, pp. 711–712. *In* **Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval**. ACM, New York, NY, USA.

Trotman, A. and B. Sigurbjörnsson. 2004a. Narrowed Extended XPath I (NEXI), pp. 16–40. *In* **Advances in XML Information Retrieval: The Third International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**, vol. 3493. Dagstuhl Castle, Germany.

Trotman, A. and B. Sigurbjörnsson. 2004b. NEXI, Now and Next, pp. 41–53. *In* **Advances in XML Information Retrieval: The Third International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**, vol. 3493. Dagstuhl Castle, Germany.

Trotman, A. and Q. Wang. 2011. Overview of the INEX 2010 Data Centric Track, pp. 171–181. *In* S. Geva, J. Kamps and A. Trotman, eds., **Comparative Evaluation of Focused Retrieval: The 9th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Verbyst, D. and P. Mulhem. 2009. Using Collectionlinks and Documents as Context for INEX 2008. *In* S. Geva, J. Kamps and A. Trotman, eds., **Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval**. pp. 87–96. Berlin, Heidelberg.

Wang, Q., Y. Gan and Y. Sun. 2012. RUC at INEX 2011 Data-Centric Track, pp. 167–179. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Wichaiwong, T. and C. Jaruskulchai. 2007. Improve XML Web Services' Performance By Compressing XML Schema tag, pp. 9–12. *In* **The 4th International Technical Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology**. Chiang Rai, Thailand.

Wichaiwong, T. and C. Jaruskulchai. 2010. A Simple Approach to Optimize XML Retrieval, pp. 426–431. *In* **The 6th International Conference on Next Generation Web Services Practices**. India.

Wichaiwong, T. and C. Jaruskulchai. 2011a. A Comparative Study Weighting Schemes for Double Scoring Technique, pp. 443–447. *In* **The World Congress on Engineering and Computer Science 2011**. San Francisco, USA.

Wichaiwong, T. and C. Jaruskulchai. 2011b. An Extended XML Compression Technique for XML Element Retrieval, pp. 539–554. *In* **The International Conference on IT Convergence and Security 2011**. Replubic of Korea.

Wichaiwong, T. and C. Jaruskulchai. 2011c. MEXIR: An Implementation of High Performance and High Precision on XML Retrieval. **Computer Technology and Application**. 2(4):301–310.

Wichaiwong, T. and C. Jaruskulchai. 2011d. XML Retrieval More Efficient Using ADXPI Indexing Scheme, pp. 638–643. *In* **International Conference on Advanced Information Networking and Applications Workshops**. IEEE Computer Society, Los Alamitos, CA, USA.

Wichaiwong, T. and C. Jaruskulchai. 2011e. XML Retrieval More Efficient Using Double Scoring Scheme, pp. 351–362. *In* **Comparative Evaluation of Focused Retrieval : The 9th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Wichaiwong, T. and C. Jaruskulchai. 2012a. A COMPRESSION TECHNIQUE FOR XML ELEMENT RETRIEVAL. **Intelligent Control and Innovative Computing Series**. 110:203–215.

Wichaiwong, T. and C. Jaruskulchai. 2012b. An Extension XML Compression Technique for XML Element Retrieval. **Information-An International Interdisciplinary Journal**. 15.

Wichaiwong, T. and C. Jaruskulchai. 2012c. MEXIR at INEX-2011, pp. 180–187. *In* **Focused Retrieval of Content and Structure: The 10th International Workshop of the Initiative for the Evaluation of XML, Springer, Lecture Notes in Computer Science**. Dagstuhl Castle, Germany.

Wichaiwong, T. and C. Jaruskulchai. 2012d. A Score Sharing Method for XML Element Retrieval. **Information-An International Interdisciplinary Journal**. 15(10):4165–4178.

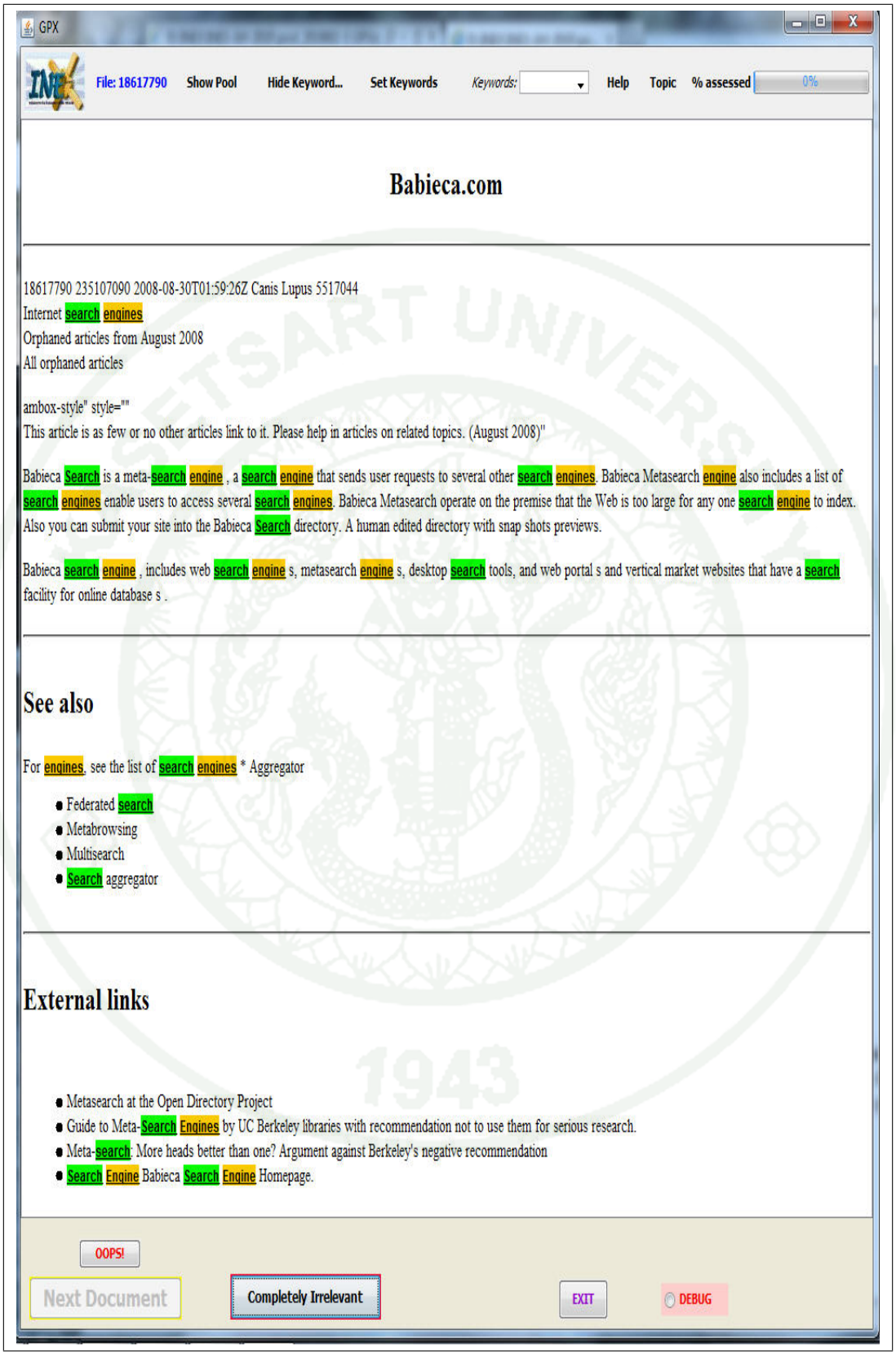
Wichaiwong, T. and C. Jaruskulchai. 2013. A Double Scoring Method for XML Element Retrieval. **Computing and Informatics**. 32:1001–1030.

Wichaiwong, T., K. Koonsanit and C. Jaruskulchai. 2008. A Simple Approach to Optimized Text Compression's Performance, pp. 66–70. *In* **Proceedings of the 2008 4th International Conference on Next Generation Web Services Practices**. IEEE Computer Society, Washington, DC, USA.

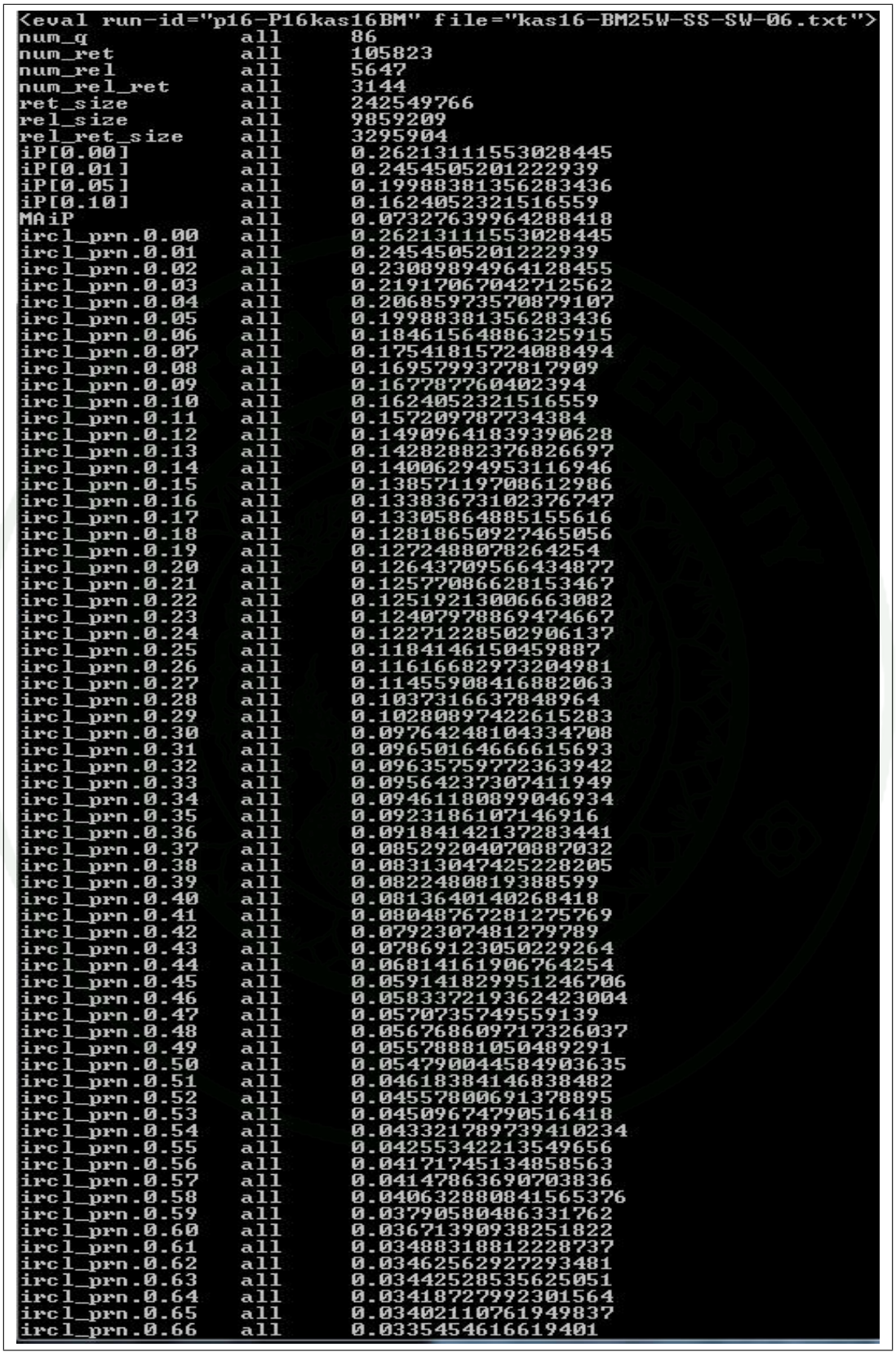
APPENDICES

Appendix A

INEX Evaluation Tools and Result



Appendix Figure A1 INEX Evaluation Tool



```
<eval run-id="p16-P16kas16BM" file="kas16-BM25W-SS-SW-06.txt">
num_q             all     86
num_ret           all     105823
num_rel           all     5647
num_rel_ret       all     3144
ret_size          all     242549766
rel_size          all     9859209
rel_ret_size      all     3295904
iP[0.00]          all     0.26213111553028445
iP[0.01]          all     0.2454505201222939
iP[0.05]          all     0.199883813562834 36
iP[0.10]          all     0.1624052321516559
MAiP              all     0.07327639964288418
ircl_prn.0.00     all     0.26213111553028445
ircl_prn.0.01     all     0.2454505201222939
ircl_prn.0.02     all     0.23089894964128455
ircl_prn.0.03     all     0.21917067042712562
ircl_prn.0.04     all     0.20685973570879107
ircl_prn.0.05     all     0.19988381356283436
ircl_prn.0.06     all     0.18461564886325915
ircl_prn.0.07     all     0.17541815724088494
ircl_prn.0.08     all     0.1695799377817909
ircl_prn.0.09     all     0.167787760402394
ircl_prn.0.10     all     0.1624052321516559
ircl_prn.0.11     all     0.1572097787734384
ircl_prn.0.12     all     0.14909641839390628
ircl_prn.0.13     all     0.14282882376826697
ircl_prn.0.14     all     0.14006294953116946
ircl_prn.0.15     all     0.13857119708612986
ircl_prn.0.16     all     0.13383673102376747
ircl_prn.0.17     all     0.13305864885155616
ircl_prn.0.18     all     0.12818650927465056
ircl_prn.0.19     all     0.1272488078264254
ircl_prn.0.20     all     0.12643709566434877
ircl_prn.0.21     all     0.1257708662815 3467
ircl_prn.0.22     all     0.12519213006663082
ircl_prn.0.23     all     0.12407978869474667
ircl_prn.0.24     all     0.12271228502906137
ircl_prn.0.25     all     0.1184146150459887
ircl_prn.0.26     all     0.116166829732049 81
ircl_prn.0.27     all     0.11455908416882063
ircl_prn.0.28     all     0.1037316637848964
ircl_prn.0.29     all     0.10280897422615283
ircl_prn.0.30     all     0.09764248104334708
ircl_prn.0.31     all     0.09650164666615693
ircl_prn.0.32     all     0.09635759772363942
ircl_prn.0.33     all     0.09564237307411949
ircl_prn.0.34     all     0.09461180899046934
ircl_prn.0.35     all     0.0923186107146916
ircl_prn.0.36     all     0.09184142137283441
ircl_prn.0.37     all     0.08529204070887032
ircl_prn.0.38     all     0.08313047425228205
ircl_prn.0.39     all     0.082248081938 8599
ircl_prn.0.40     all     0.08136401402 68418
ircl_prn.0.41     all     0.08048767281275769
ircl_prn.0.42     all     0.0792307481279789
ircl_prn.0.43     all     0.07869123050229264
ircl_prn.0.44     all     0.06814161906764254
ircl_prn.0.45     all     0.05914182995124 6706
ircl_prn.0.46     all     0.05833721936242 3004
ircl_prn.0.47     all     0.0570735749559139
ircl_prn.0.48     all     0.05676860971732 6037
ircl_prn.0.49     all     0.05578881050489291
ircl_prn.0.50     all     0.05479004458490 3635
ircl_prn.0.51     all     0.04618384146838482
ircl_prn.0.52     all     0.04557800691378895
ircl_prn.0.53     all     0.04509674790516418
ircl_prn.0.54     all     0.04332178973941 0234
ircl_prn.0.55     all     0.04255342213549656
ircl_prn.0.56     all     0.04171745134858563
ircl_prn.0.57     all     0.04147863690703836
ircl_prn.0.58     all     0.04063288084156 5376
ircl_prn.0.59     all     0.03790580486331762
ircl_prn.0.60     all     0.03671390938251822
ircl_prn.0.61     all     0.03488318812228737
ircl_prn.0.62     all     0.03462562927293481
ircl_prn.0.63     all     0.03442528535625051
ircl_prn.0.64     all     0.03418727992301564
ircl_prn.0.65     all     0.03402110761949837
ircl_prn.0.66     all     0.0335454616619401
```

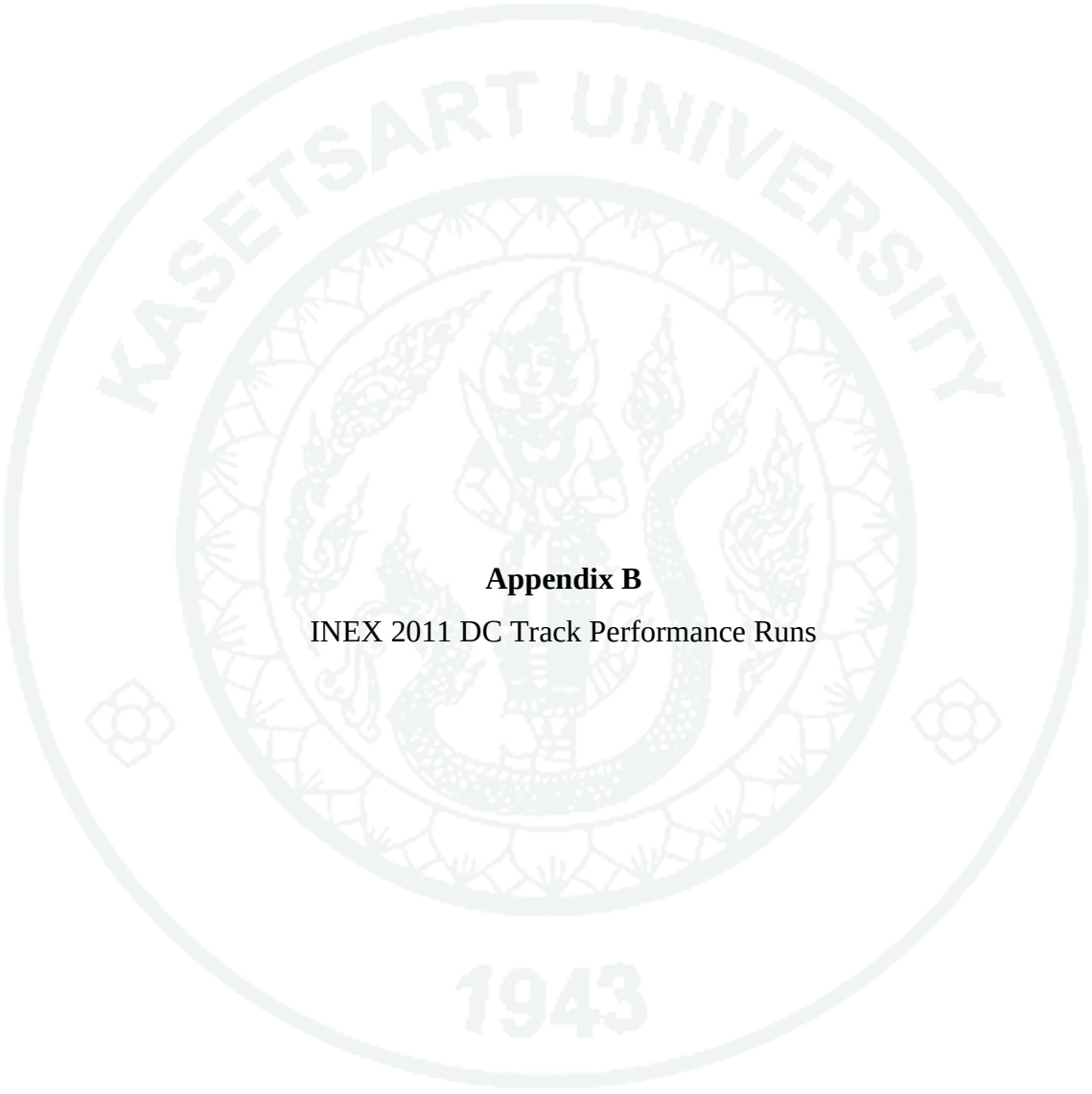
Appendix Figure A2 INEX Evaluation Result

```
<topic id="679" ct_no="4">
  <title>london</title>
  <castitle>//*[about(., london)]</castitle>
  <description>london</description>
  <narrative>london</narrative>
</topic>
<topic id="680" ct_no="13">
  <title>venice</title>
  <castitle>//*[about(., venice)]</castitle>
  <description>venice</description>
  <narrative>venice</narrative>
</topic>
<topic id="681" ct_no="16">
  <title>pokemon</title>
  <castitle>//*[about(., pokemon)]</castitle>
  <description>pokemon</description>
  <narrative>pokemon</narrative>
</topic>
<topic id="682" ct_no="20">
  <title>kaikorai valley college</title>
  <castitle>//*[about(., kaikorai valley college)]</castitle>
  <description>kaikorai valley college</description>
  <narrative>kaikorai valley college</narrative>
</topic>
<topic id="683" ct_no="41">
  <title>contraceptive pill</title>
  <castitle>//*[about(., contraceptive pill)]</castitle>
  <description>contraceptive pill</description>
  <narrative>contraceptive pill</narrative>
</topic>
<topic id="684" ct_no="59">
  <title>saratoga spa</title>
  <castitle>//*[about(., saratoga spa)]</castitle>
  <description>saratoga spa</description>
  <narrative>saratoga spa</narrative>
</topic>
<topic id="685" ct_no="60">
  <title>rugby world cup</title>
  <castitle>//*[about(., rugby world cup)]</castitle>
  <description>rugby world cup</description>
  <narrative>rugby world cup</narrative>
</topic>
<topic id="686" ct_no="64">
  <title>uv filter</title>
  <castitle>//*[about(., uv filter)]</castitle>
  <description>uv filter</description>
  <narrative>uv filter</narrative>
</topic>
```

Appendix Figure A3 An example of an INEX Ad hoc Topic

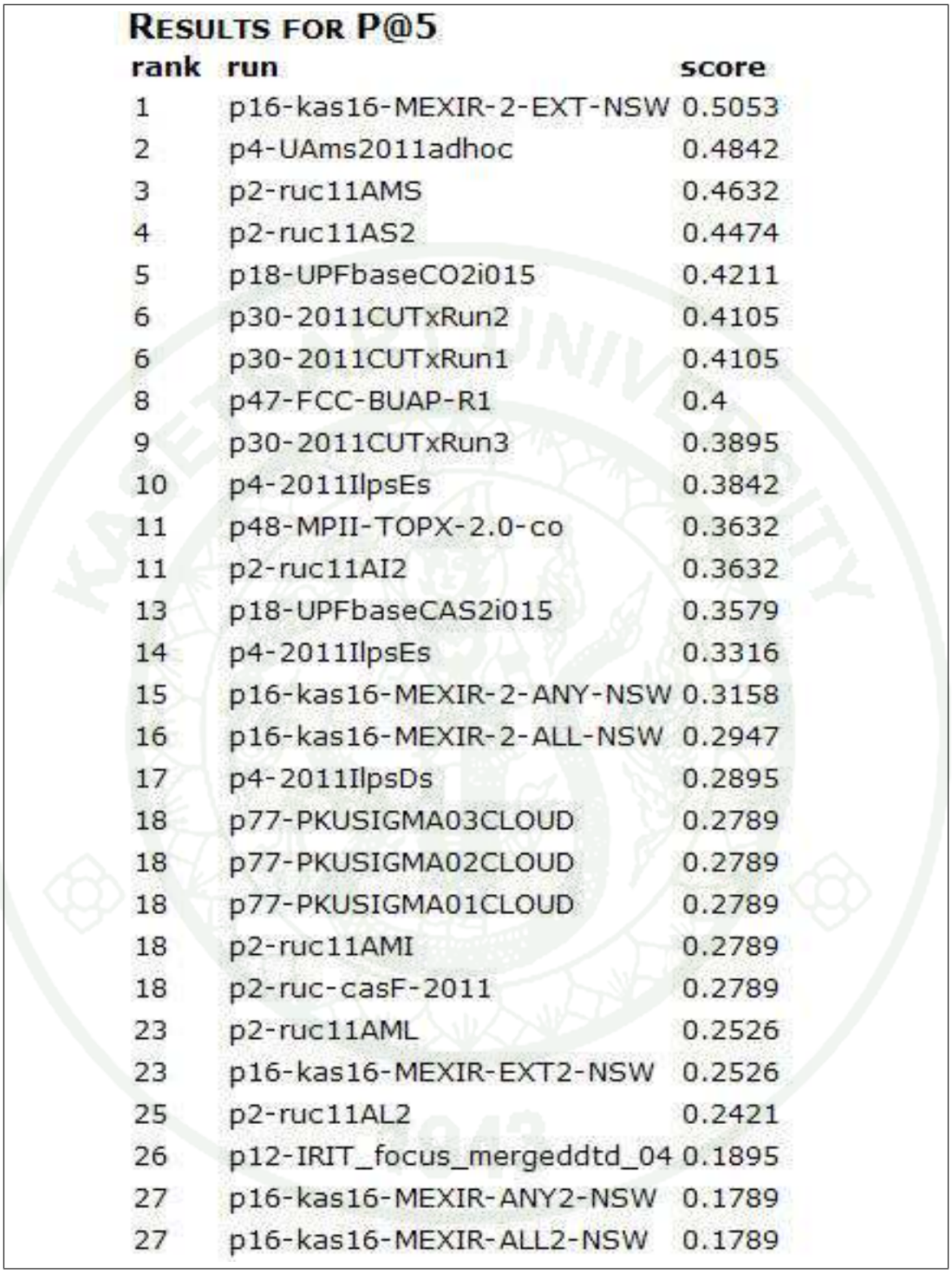
```
<topic id="2011123" guid="55">
    <task>AdHoc</task>
    <title>Worst actor century</title>
    <castitle>//person[about(.//trivia, Worst Actor of the Century)]//name</castitle>
    <description>Searching for the personed awarded "worst actor of the century".</description>
    <narrative>I am looking for the personed awarded "the worst actor of the century".</narrative>
</topic>
<topic id="2011124" guid="57">
    <task>AdHoc</task>
    <title>survive desert island</title>
    <castitle>//movie[about(., survive desert island)]</castitle>
    <description>I like to find movies or documentaries about surviving on a desert island</description>
    <narrative>I am a teacher and I want to teach my students survival techniques.
    Any movie or documentary about the topic is relevant. I am not interested in TV programs or reality shows.</narrative>
</topic>
<topic id="2011125" guid="59">
    <task>AdHoc</task>
    <title>Movies directed by Steven Spielberg</title>
    <castitle>//movie[about(.//director, Steven Spielberg)]//title</castitle>
    <description>Name all movies directed by Steven Spielberg</description>
    <narrative>List the names of all movies that have been directed by Steven Spielberg so far.</narrative>
</topic>
<topic id="2011126" guid="46">
    <task>AdHoc</task>
    <title>German fellow actors of Mel Gibson</title>
    <castitle>//actor[about(.//name, Mel Gibson ) and about(.//biography, Germany)]</castitle>
    <description>I want to find all German fellow actors of Mel Gibson.</description>
    <narrative>I want to know the names of German actors that played in movies together with Mel Gibson.</narrative>
</topic>
<topic id="2011127" guid="43">
    <task>AdHoc</task>
    <title>Maureen Lipman as mother of Jewish musician</title>
    <castitle>//movie[about(.//actor, Maureen Lipman ) and about(.//character, mother) and about(.//character, musician)]</castitle>
    <description>What was the title of the movie where Maureen Lipman played the mother of a Polish Jewish musician</description>
    <narrative>I want to know original title of the movie where Maureen Lipman played the mother of a Polish Jewish musician.</narrative>
</topic>
<topic id="2011128" guid="56">
    <task>AdHoc</task>
    <title>lovers arctic Spanish</title>
    <castitle>//movie[about(.//title, arctic lovers) AND about(.//additional_details, Spanish)]</castitle>
    <description>I am searching for a Spanish movie which title was something about arctic lovers</description>
    <narrative>I saw this movie long time ago and I would like to watch it again.</narrative>
</topic>
<topic id="2011129" guid="51">
    <task>AdHoc</task>
    <title>Alien Movies before 1970</title>
    <castitle>//movie[about(.//plot, Alien) AND .//releasedate &lt; 1970]</castitle>
    <description>I want to find movies about extraterrestrials that were shot before 1970.</description>
    <narrative>I want to find movies about extraterrestrials that were shot before 1970.</narrative>
</topic>
```

Appendix Figure A4 The Example of INEX Data Centric Topic



Appendix B

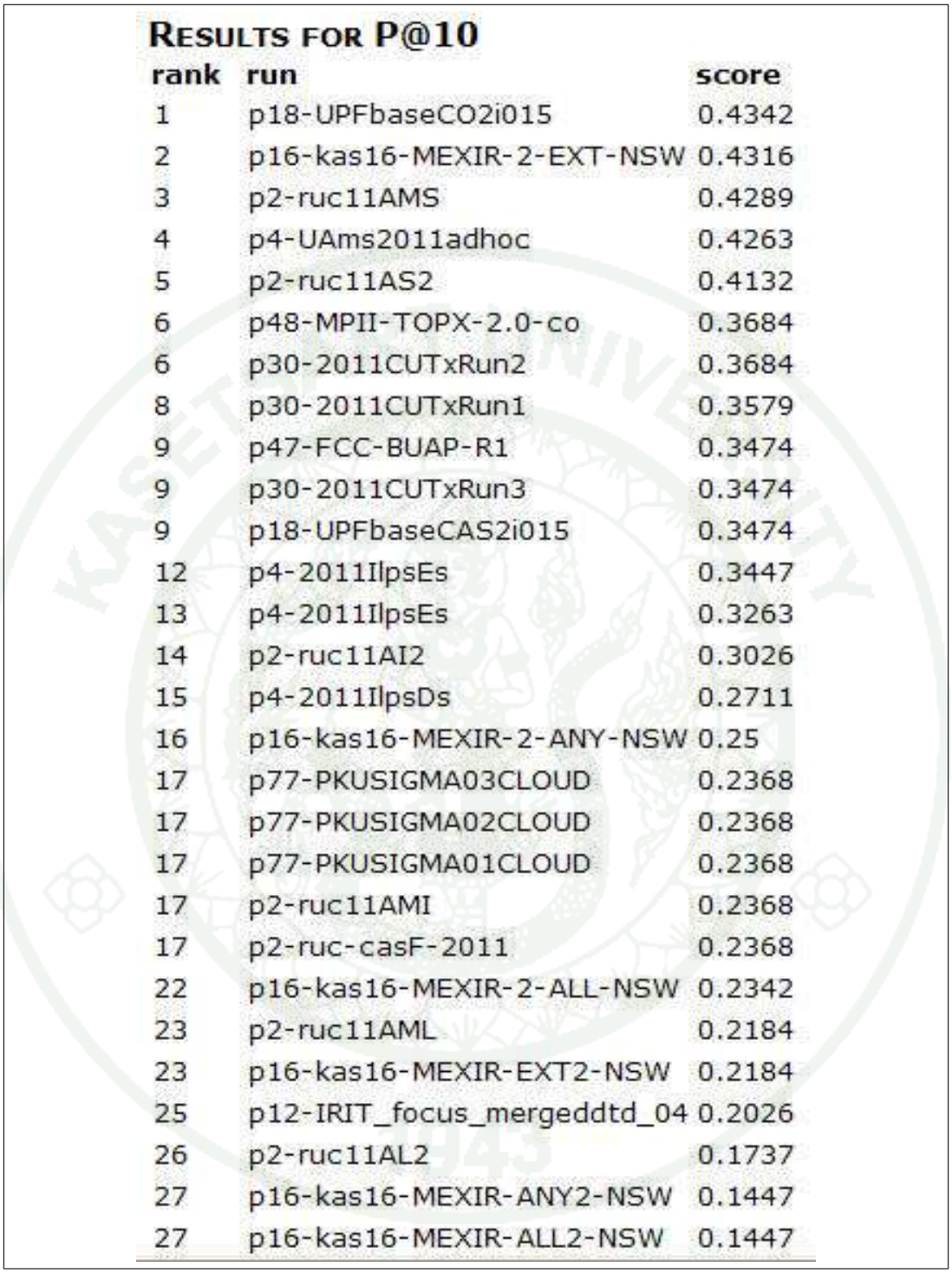
INEX 2011 DC Track Performance Runs



RESULTS FOR P@5

rank	run	score
1	p16-kas16-MEXIR-2-EXT-NSW	0.5053
2	p4-UAms2011adhoc	0.4842
3	p2-ruc11AMS	0.4632
4	p2-ruc11AS2	0.4474
5	p18-UPFbaseCO2i015	0.4211
6	p30-2011CUTxRun2	0.4105
6	p30-2011CUTxRun1	0.4105
8	p47-FCC-BUAP-R1	0.4
9	p30-2011CUTxRun3	0.3895
10	p4-2011IlpsEs	0.3842
11	p48-MPII-TOPX-2.0-co	0.3632
11	p2-ruc11AI2	0.3632
13	p18-UPFbaseCAS2i015	0.3579
14	p4-2011IlpsEs	0.3316
15	p16-kas16-MEXIR-2-ANY-NSW	0.3158
16	p16-kas16-MEXIR-2-ALL-NSW	0.2947
17	p4-2011IlpsDs	0.2895
18	p77-PKUSIGMA03CLOUD	0.2789
18	p77-PKUSIGMA02CLOUD	0.2789
18	p77-PKUSIGMA01CLOUD	0.2789
18	p2-ruc11AMI	0.2789
18	p2-ruc-casF-2011	0.2789
23	p2-ruc11AML	0.2526
23	p16-kas16-MEXIR-EXT2-NSW	0.2526
25	p2-ruc11AL2	0.2421
26	p12-IRIT_focus_mergeddtd_04	0.1895
27	p16-kas16-MEXIR-ANY2-NSW	0.1789
27	p16-kas16-MEXIR-ALL2-NSW	0.1789

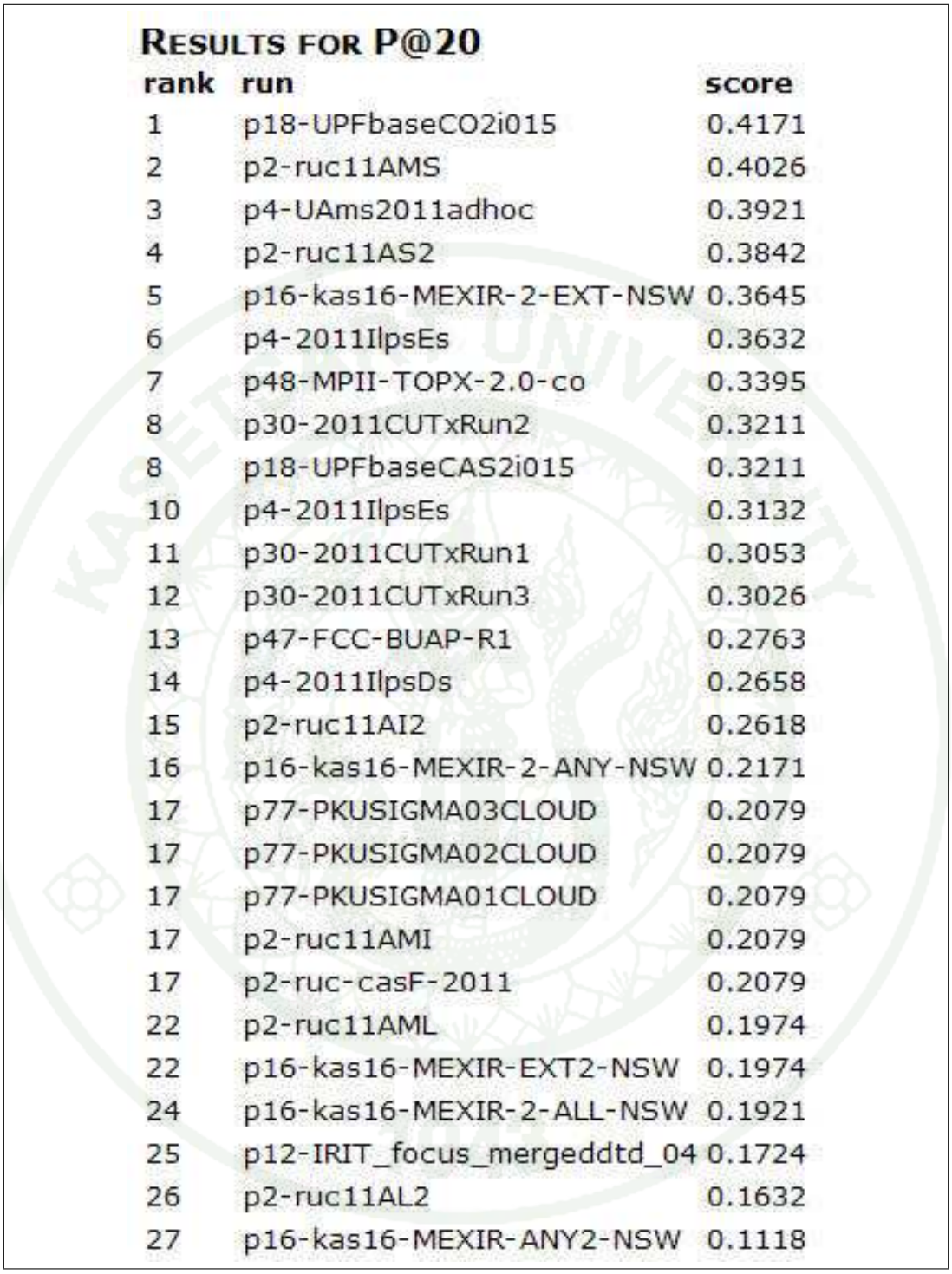
Appendix Figure B1 INEX 2011 DC Track - Results for P@5



RESULTS FOR P@10

rank	run	score
1	p18-UPFbaseCO2i015	0.4342
2	p16-kas16-MEXIR-2-EXT-NSW	0.4316
3	p2-ruc11AMS	0.4289
4	p4-UAms2011adhoc	0.4263
5	p2-ruc11AS2	0.4132
6	p48-MPII-TOPX-2.0-co	0.3684
6	p30-2011CUTxRun2	0.3684
8	p30-2011CUTxRun1	0.3579
9	p47-FCC-BUAP-R1	0.3474
9	p30-2011CUTxRun3	0.3474
9	p18-UPFbaseCAS2i015	0.3474
12	p4-2011IlpsEs	0.3447
13	p4-2011IlpsEs	0.3263
14	p2-ruc11AI2	0.3026
15	p4-2011IlpsDs	0.2711
16	p16-kas16-MEXIR-2-ANY-NSW	0.25
17	p77-PKUSIGMA03CLOUD	0.2368
17	p77-PKUSIGMA02CLOUD	0.2368
17	p77-PKUSIGMA01CLOUD	0.2368
17	p2-ruc11AMI	0.2368
17	p2-ruc-casF-2011	0.2368
22	p16-kas16-MEXIR-2-ALL-NSW	0.2342
23	p2-ruc11AML	0.2184
23	p16-kas16-MEXIR-EXT2-NSW	0.2184
25	p12-IRIT_focus_mergeddtd_04	0.2026
26	p2-ruc11AL2	0.1737
27	p16-kas16-MEXIR-ANY2-NSW	0.1447
27	p16-kas16-MEXIR-ALL2-NSW	0.1447

Appendix Figure B2 INEX 2011 DC Track - Results for P@10



RESULTS FOR P@20

rank	run	score
1	p18-UPFbaseCO2i015	0.4171
2	p2-ruc11AMS	0.4026
3	p4-UAms2011adhoc	0.3921
4	p2-ruc11AS2	0.3842
5	p16-kas16-MEXIR-2-EXT-NSW	0.3645
6	p4-2011IlpsEs	0.3632
7	p48-MPII-TOPX-2.0-co	0.3395
8	p30-2011CUTxRun2	0.3211
8	p18-UPFbaseCAS2i015	0.3211
10	p4-2011IlpsEs	0.3132
11	p30-2011CUTxRun1	0.3053
12	p30-2011CUTxRun3	0.3026
13	p47-FCC-BUAP-R1	0.2763
14	p4-2011IlpsDs	0.2658
15	p2-ruc11AI2	0.2618
16	p16-kas16-MEXIR-2-ANY-NSW	0.2171
17	p77-PKUSIGMA03CLOUD	0.2079
17	p77-PKUSIGMA02CLOUD	0.2079
17	p77-PKUSIGMA01CLOUD	0.2079
17	p2-ruc11AMI	0.2079
17	p2-ruc-casF-2011	0.2079
22	p2-ruc11AML	0.1974
22	p16-kas16-MEXIR-EXT2-NSW	0.1974
24	p16-kas16-MEXIR-2-ALL-NSW	0.1921
25	p12-IRIT_focus_mergeddtd_04	0.1724
26	p2-ruc11AL2	0.1632
27	p16-kas16-MEXIR-ANY2-NSW	0.1118

Appendix Figure B3 INEX 2011 DC Track - Results for P@20

RESULTS FOR P@30

rank	run	score
1	p2-ruc11AMS	0.3877
2	p18-UPFbaseCO2i015	0.3825
3	p2-ruc11AS2	0.3684
4	p4-UAms2011adhoc	0.3579
5	p4-2011IlpsEs	0.3404
6	p16-kas16-MEXIR-2-EXT-NSW	0.3298
7	p48-MPII-TOPX-2.0-co	0.3289
8	p18-UPFbaseCAS2i015	0.307
9	p4-2011IlpsEs	0.3035
10	p30-2011CUTxRun2	0.2965
11	p30-2011CUTxRun3	0.2851
12	p30-2011CUTxRun1	0.2798
13	p4-2011IlpsDs	0.2702
14	p2-ruc11AI2	0.2474
15	p47-FCC-BUAP-R1	0.2412
16	p77-PKUSIGMA03CLOUD	0.1956
16	p77-PKUSIGMA02CLOUD	0.1956
16	p77-PKUSIGMA01CLOUD	0.1956
16	p2-ruc11AMI	0.1956
16	p2-ruc-casF-2011	0.1956
21	p2-ruc11AML	0.1939
21	p16-kas16-MEXIR-EXT2-NSW	0.1939
23	p16-kas16-MEXIR-2-ANY-NSW	0.193
24	p12-IRIT_focus_mergeddtd_04	0.1702
25	p16-kas16-MEXIR-2-ALL-NSW	0.1675
26	p2-ruc11AL2	0.1632
27	p16-kas16-BM25W-NSS-SW	0.107

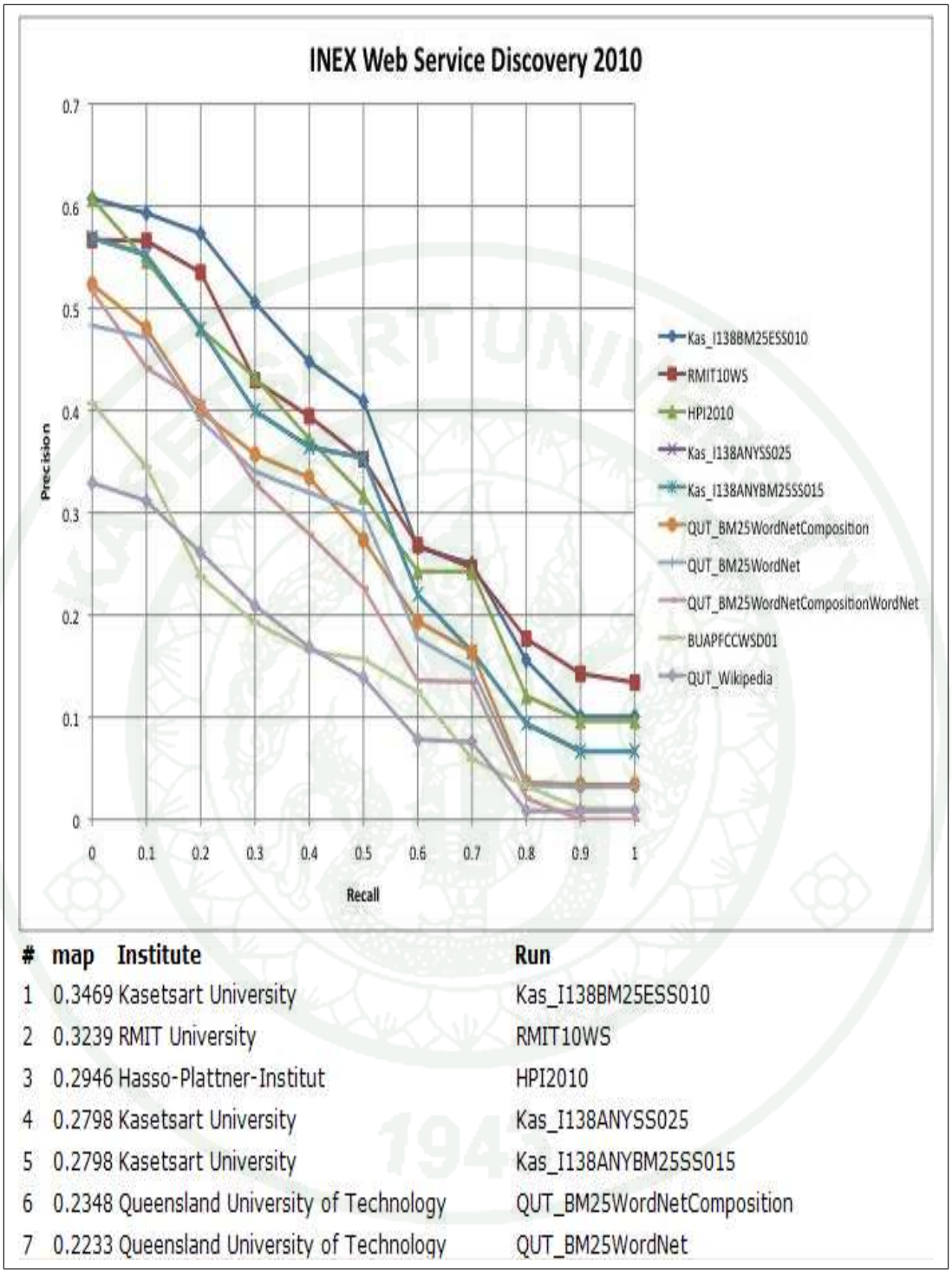
Appendix Figure B4 INEX 2011 DC Track - Results for P@30

Appendix C

INEX 2010 Web Service Track Performance Runs

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns:witw="http://norman.walsh.name/witw"
             name="WITW"
             targetNamespace="http://norman.walsh.name/witw">

  <message name="NameRequest">
    <part name="userid" type="xsd:string"/>
  </message>

  <message name="NameResponse">
    <part name="name" type="xsd:string"/>
  </message>

  <portType name="NamePortType">
    <operation name="name">
      <input message="witw:NameRequest"/>
      <output message="witw:NameResponse"/>
    </operation>
  </portType>

  <binding name="NameBinding" type="witw:NamePortType">
    <soap:binding style="rpc"
          transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="name">
      <soap:operation soapAction="http://norman.walsh.name/witw#name"/>
      <input>
    <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://norman.walsh.name/witw"
        use="encoded"/>
      </input>
      <output>
    <soap:body
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
        namespace="http://norman.walsh.name/witw"
        use="encoded"/>
      </output>
    </operation>
  </binding>

  <service name="WitwService">
    <documentation>WSDL File for WITW Services</documentation>
    <port binding="witw:NameBinding" name="NamePortType">
      <soap:address location="http://norman.walsh.name/2005/02/soap/witw"/>
    </port>
  </service>
</definitions>
```

Appendix Figure C1 The Example Document of INEX-2010 Web Service Discovery Track



#	map	Institute	Run
1	0.3469	Kasetsart University	Kas_I138BM25ESS010
2	0.3239	RMIT University	RMIT10WS
3	0.2946	Hasso-Plattner-Institut	HPI2010
4	0.2798	Kasetsart University	Kas_I138ANYSS025
5	0.2798	Kasetsart University	Kas_I138ANYBM25SS015
6	0.2348	Queensland University of Technology	QUT_BM25WordNetComposition
7	0.2233	Queensland University of Technology	QUT_BM25WordNet

Appendix Figure C2 Performance of 10 best INEX-2010 Web Service Discovery Results

CURRICULUM VITAE

NAME : Mr. Tanakorn Wichaiwong

BIRTH DATE : April 27, 1977

BIRTH PLACE : Udonthani, Thailand

EDUCATION :

YEAR	INSTITUTE	DEGREE/DIPLOMA
2001	North Eastern University	B.B.A. (Business Computer)
2008	Kasetsart University	M.S. (Computer Science)

POSITION/TITLE : SNR SYSTEMS ANALYST

WORK PLACE : STATS ChipPAC Services (Thailand) Ltd.

SCHOLARSHIP/AWARDS : ICITCS-2011 Best Paper Award, CUPT-2011 Award