

# Impact of Terminal Node Processing on Element Retrieval

A thesis submitted to the faculty of the graduate school of the University of Minnesota by

Vikram Malik

in partial fulfillment of the requirements for the degree of Master of Science

August 2007

Department of Computer Science  
University of Minnesota Duluth  
Duluth, MN 55812  
USA

UNIVERSITY OF MINNESOTA

This is to certify that I have examined this copy of master's thesis by

Vikram Malik

and have found that it is complete and satisfactory in all respects,  
and that any and all revisions required by the final  
examining committee have been made.

Carolyn J. Crouch

---

Name of Faculty Advisor

Signature of Faculty Advisor

Date

GRADUATE SCHOOL

© Vikram Malik 2007

## Acknowledgements

First and foremost, I would like to thank my advisor Dr Carolyn Crouch, for her support and guidance for the last two years. She was patient and always willing to share her immense experience in the field.

I would also like to thank Dr Donald Crouch and Steve Holtz for their valuable suggestions and feedback time and again.

I would like to thank Jim Luttinen for promptly taking care of the system failures we had this year, even while he was on vacation.

I would also like to thank Lori Lucia and Linda Meek for all their administrative help and words of encouragement.

Finally, I would like to thank the Department of Computer Science and my friends here in Duluth for making the last two years an unforgettable experience for me.

## Abstract

Information retrieval strategies of earlier years were developed around the idea of retrieving entire documents in response to a user's request. But with widespread use of markup languages like XML (Extensible Markup Language) for representing documents, the idea of retrieving at the element level evolved. And with the continuous growth in XML information repositories available today, the focus on developing effective element retrieval strategies has never been stronger.

Our element retrieval strategy, called *flexible retrieval* [7], requires maintaining only a single index of the leaf nodes of documents and is designed to work with a strictly structured collection of well-defined and non-overlapping element nodes. In this research, we examine the feasibility of extending this technique to the real-world, semi-structured Wikipedia collection [4]. We tune for slope and pivot values for this collection against the new set of cumulative gain based measures [6] adopted by INEX in 2007. Experiments comparing the results obtained from our technique with retrieval done against an overlapping index of all document elements (the *all-element index*) are detailed. We describe experiments to determine the impact of processing terminal nodes prior to evaluation and show that it dramatically improves evaluation results in most cases.

# Table of Contents

1. Introduction.....	6
2. Overview.....	8
2.1 INEX (INitiative for Evaluation of XML retrieval) .....	8
2.2 Wikipedia Collection .....	8
2.3 Topic Collection.....	9
2.4 Relevance Assessments .....	12
2.5 The Smart Information Retrieval System .....	14
2.6 Document Length Normalization .....	14
3. Retrieval Tasks and Evaluation Metrics .....	17
3.1 Retrieval Tasks - 2006 .....	17
3.2 Retrieval Tasks - 2007 .....	18
3.3 Evaluation Metrics .....	18
4. Our Approach.....	21
4.1 Untagged Text.....	23
4.2 Parsing.....	23
4.3 Smart Retrieval .....	23
4.4 Flex Retrieval/Dynamic Retrieval .....	24
4.5 The Weighting Scheme.....	25
5. Experiments and Results.....	26
5.1 Experiments .....	26
Terminal Node Expansion .....	26
Structural Constraints on Queries (CAS).....	27
5.2 Results.....	30
Thorough subtask:.....	30
Focused sub-task.....	33
Relevant In Context (RIC) sub-task.....	37
Best In Context (BIC) sub-task.....	40
6. Conclusions and Future Work .....	42
References.....	43

## List of Figures

Figure 1	A sample CO+S topic.....	11
Figure 2	Rel-assessment 289.xml .....	13
Figure 3	Pivoted document length normalization [10] .....	15
Figure 4	<i>Lnu</i> element term weighting formula .....	16
Figure 5	Itu query term weighting formula [10] .....	16
Figure 6	recall and precision measures .....	18
Figure 7	Flow diagram for all-element retrieval.....	21
Figure 8	Flow diagram for flexible retrieval.....	22
Figure 9	Doctree for an XML document [7].....	23
Figure 10	Merging <i>nnn</i> vectors to generate inner node vectors [7].....	24
Figure 11	Original retrieval and post-processed retrieval.....	27

## List of Tables

Table 1	Wikipedia vs IEEE.....	9
Table 2	Wikipedia collection statistics .....	9
Table 3	INEX 2006 Topic Components .....	10
Table 4	INEX 2007 Topic Components .....	10
Table 5	Terminal Nodes.....	26
Table 6	INEX 2006 queries and structural constraints .....	28
Table 7	INEX 2007 queries and structural constraints. ....	29
Table 8.1	Thorough CO with no terminal node expansion (MAep).....	31
Table 8.2	Thorough CO with terminal node expansion (MAep).....	32
Table 8.3	Thorough CAS with no terminal node expansion (MAep).....	32
Table 8.4	Thorough CAS with terminal node expansion (MAep).....	33
Table 9. 1	Focused (overlap OFF) CO with no terminal node expansion (nxCG) .....	34
Table 9. 2	Focused (overlap OFF) CO with terminal node expansion (nxCG).....	34
Table 9. 3	Focused (overlap ON) CO with no node expansion (nxCG).....	35
Table 9. 4	Focused (overlap ON) CO with terminal node processing (nxCG).....	35
Table 9. 5	Focused (overlap OFF) CAS with no terminal node expansion (nxCG).....	36
Table 9. 6	Focused (overlap OFF) CAS with terminal node expansion (nxCG).....	36
Table 9. 7	Focused (overlap ON) CAS with no terminal node expansion (nxCG) .....	37
Table 9. 8	Focused (overlap ON) CAS with terminal node expansion (nxCG) .....	37
Table 10. 1	Relevant In Context CO with no terminal node expansion (gP) .....	38
Table 10. 2	Relevant In Context CO with terminal node expansion (gP) .....	38
Table 10. 3	Relevant In Context CAS with no terminal node expansion (gP) .....	39
Table 10. 4	Relevant In Context CAS with terminal node expansion (gP) .....	39
Table 11. 1	BEP CO with no terminal node expansion (BEPD) .....	40
Table 11. 2	BEP CO with terminal node expansion (BEPD) .....	40
Table 11. 3	BEP CAS with no terminal node expansion (BEPD) .....	41
Table 11. 4	BEP CAS with terminal node expansion (BEPD) .....	41

# 1. Introduction

With the vast collections of data that are now available in electronic form, searching through this data to locate relevant information is a key field of research today. Information retrieval is the branch of science that deals with the design of strategies to search for relevant information in large collections of data.

Traditional information retrieval strategies focused on retrieving information from unstructured, ordinary documents of text. A set of documents identified as relevant was returned as the result of the search. But with the advent of markup languages like XML, more and more information is now stored in structured form. As a result of this change, the focus of information retrieval is now shifting to design of strategies that make use of the structure of documents to return specific document elements, rather than complete documents.

XML markup of data allows a document to be partitioned into well-defined and non-overlapping elements. These elements may contain either actual text of the document or may only provide additional semantic information about its content. Structuring a document in this manner gives it a tree-like structure and enables searching for relevant information at various levels of this tree to then return a set of highly correlating elements to the user. We call this approach *flexible retrieval* [7] or, alternatively, *dynamic element retrieval*.

Our approach, which is detailed in Chapter 3, involves creating an index of only the leaf nodes. The leaf is the smallest meaningful element in a document, usually a paragraph. Retrieval is then carried out on this index to generate a set of highly correlating leaf nodes. We use the Smart [9] retrieval engine and the Vector Space Model (detailed later) to carry out these indexing and retrieval tasks. The retrieval output is fed as input to our system *Flex*, a retrieval engine we developed here at the University of Minnesota Duluth. *Flex* then builds the inner nodes (vectors) of the documents to which these retrieved leaf nodes belong, weights the query as well as the generated vectors, and computes the similarity between the two. This vector comparison is at the kernel of our approach and results in a list of elements ranked by their similarity scores. We also create an index,

using Smart, of each identified element in each document. This results in an index of overlapping document elements, which we call the *all-element index*. Retrieval is then performed against this index. This retrieval is used as the base case against which we compare the results obtained by Flex.

Structured retrieval systems allow a user to refer explicitly to document structure within the queries. For instance, a user can ask that only paragraphs or sections be returned in response to his queries. We post-process Flex results to fulfill this requirement, as explained in Chapter 4.

The primary focus of this work is to extend the flexible retrieval system (Flex) developed here at UMD to the new, real-world and semi-structured Wikipedia collection. Unlike the structured IEEE data that we worked with the previous year (2006), the Wikipedia collection poses a new challenge due to its structural inconsistencies and sheer size. Chapter 2 lays the necessary groundwork and gives an overview of INEX [8], collection characteristics, topics, relevance-assessments, the Smart retrieval system and document length normalization. Chapter 3 describes retrieval tasks and the evaluation metrics for 2006 and 2007. Our approach to structured retrieval and its implementation details are given in Chapter 4. Chapter 5 describes experiments carried out to tune collection specific parameters and to post-process the results obtained using these tuned parameters, experiments comparing our results to the all-element index results, and their analysis. Finally, Chapter 6 provides conclusions and suggestions for future work.

## 2. Overview

This section provides an overview of INEX, the test collection used for this research, topics/queries and their format, relevance assessments and the Smart retrieval system.

### 2.1 INEX (INitiative for Evaluation of XML retrieval)

INEX [8] is an initiative that facilitates the development of effective XML-based information retrieval strategies. Since its inception in 2002, INEX has provided the necessary infrastructure in the form of an XML collection for experimentation and appropriate scoring metrics for evaluation of content-oriented XML retrieval systems. UMD participates in the *Ad hoc* task of INEX. *Ad hoc* retrieval is described as a simulation of operations in a library consisting of XML documents. The task involves searching a set of static XML documents using new topics or queries. The topics may contain both content and structural constraints, and the retrieval system may return arbitrary XML elements from the library.

### 2.2 Wikipedia Collection

INEX provided a new collection this year for our experiments. This collection, based on the English language Wikipedia collection, replaces the IEEE collection used up to this point. The Wikipedia collection is different from our earlier collection in various aspects. It is only semi-structured and contains untagged text. And it is more than 10 times the size of the IEEE collection. Since our approach is based on building parent vectors from child vectors, we identify and markup the untagged text (as described further in chapter 4). Table 1 shows some of the major differences between the IEEE and the Wikipedia collections that have been made available by INEX.

Table 1 Wikipedia vs IEEE

	<b>IEEE (2006)</b>	<b>WIKIPEDIA (2007)</b>
<i>Size</i>	494 MB	5.8 GB
<i>No. of docs</i>	12,107	659,388
<i>Provided by</i>	IEEE computer society	wikipedia.org

We computed various statistics for the Wikipedia collection and found that it contains 52,555,826 elements with 1258 different tag names for them. Of these, about 800 tags occur just once and add semantic structure to the content. Another 100 do not occur more than 10 times in the entire collection. Collection statistics are reported in Table 2.

Table 2 Wikipedia collection statistics

<b>Statistic</b>	<b>Count</b>
<i>No of articles</i>	659,388
<i>No of elements</i>	52,555,826
<i>No of unique tags</i>	1258
<i>Avg # of nodes per article</i>	161.35
<i>Avg depth of an element</i>	6.72

### 2.3 Topic Collection

Until 2005, the INEX topic specification consisted of two types of topics:-

- Content-Only (CO) topics: topics that ignore document structure and focus only on the information need, and
- Content and Structure (CAS) topics: topics that contain explicit references to XML structure

However, for 2006 & 2007, these topic types were merged into one: Content-Only + Structure (CO+S) topics. CO+S topics are created and then assessed by the participants at the beginning of each year. The 2006 topic collection contains 125 CO+S topics. The 2007 CO+S topic set has a total of 127 CO+S topics. An overview of the various components of the topics from both years is provided in Tables 3 and 4.

Table 3 INEX 2006 Topic Components

Component	Description
<i>&lt;title&gt;</i>	in which Content Only (CO) queries are given
<i>&lt;castitle&gt;</i>	in which Content And Structure (CAS) queries are given
<i>&lt;description&gt;</i>	a one or two sentence natural language definition of the information need
<i>&lt;narrative&gt;</i>	in which the definitive definition of relevance and irrelevance are given
<i>&lt;ontopic_keywords&gt;</i>	in which terms that are expected in relevant elements are listed
<i>&lt;offtopic_keywords&gt;</i>	in which terms that are expected in non-relevant elements are listed

Table 4 INEX 2007 Topic Components

Component	Description
<i>&lt;title&gt;</i>	in which Content Only (CO) queries are given
<i>&lt;castitle&gt;</i>	in which Content And Structure (CAS) queries are given
<i>&lt;description&gt;</i>	a one or two sentence natural language definition of the information need
<i>&lt;narrative&gt;</i>	in which the definitive definition of relevance and irrelevance are given

We use the *<title>* part of the CO+S topic for Content Only retrievals and the *<castitle>* part for retrievals that take structural constraints into account. Figure 1 contains an example topic from the 2007 INEX topic collection:

```
<Author>Vikram Malik</Author>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE inex_topic SYSTEM "topic.dtd">

<inex_topic query_type="CO+S" ct_no="35">

<InitialTopicStatement>
I would like to know about Steganography and its current techniques.
</InitialTopicStatement>

<title>
Steganography and its techniques
</title>

<castitle>
//article[about(.,Steganography) or about(.,"Steganography Techniques")]
</castitle>

<description>
Find me information about the science of Steganography, and its various
techniques used today
.</description>

<narrative>
I have to write a report on Steganography, and am interested in knowing more
about this science and how it has evolved over time. Elements detailing
techniques of Steganography will be relevant to my requirement. I would also be
interested in knowing about the real-world applications of these techniques.
</narrative>

</inex_topic>
```

Figure 1 A sample CO+S topic

## 2.4 Relevance Assessments

Relevance assessments are produced by manual assessment of every document element in the collection against every query. Each group is assigned a set of queries to assess [5]. The assessments, once completed by all the participants, are pooled to form a set of all documents found relevant to a query by the human assessors. This set of relevant document elements forms the ideal retrieval set for a query. The results produced by a retrieval system are assessed against this ideal set.

INEX provided a new online relevance assessment system (XML Retrieval Assessment Interface) for carrying out the assessments. In addition to marking a complete element as relevant, a user could also mark only part of an element. For every document that had one or more elements marked (or partially marked) as relevant, the user also specifies a Best Entry Point (BEP). BEP is that point in the document where the reader should begin reading to get the best response to this query. Results for this task consist of one xpath (its best entry point) per retrieved document for each query.

Previously, relevance of an element was defined in two dimensions - *specificity* and *exhaustivity*. *Specificity* describes the extent to which a document element focuses on the topic of interest. And *exhaustivity* measures the extent to which the document component discusses the topic. But this year the *exhaustivity* measure was dropped from assessments, and every element marked relevant is by default given an *exhaustivity* value of 2. The *specificity* of any (completely or partially highlighted) element can then be calculated as some function of the contained relevant and irrelevant content. Figure 2 shows a sample relevance assessment file for a query.

```

<file collection="wikien" name="32908">

<best-entry-point
path="/article[1]/body[1]/section[3]/p[2]/text()[10].0"/>

<passage start="/article[1]/body[1]/section[3]/p[2]/text()[10].0"
end="/article[1]/body[1]/section[3]/p[2]/text()[17].2" size="311"/>

<element path="/article[1]/body[1]/section[3]/p[2]"
exhaustivity="2" size="729" rsize="311"/>
  <element path="/article[1]/body[1]/section[3]" exhaustivity="2"
size="9923" rsize="311"/>
    <element path="/article[1]/body[1]" exhaustivity="2"
size="36056" rsize="311"/>
      <element path="/article[1]" exhaustivity="2" size="36066"
rsize="311"/>
        <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[10]"
exhaustivity="2" size="11" rsize="11"/>
          <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[11]"
exhaustivity="2" size="7" rsize="7"/>
            <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[12]"
exhaustivity="2" size="18" rsize="18"/>
              <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[13]"
exhaustivity="2" size="22" rsize="22"/>
                <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[14]"
exhaustivity="2" size="7" rsize="7"/>
                  <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[15]"
exhaustivity="2" size="17" rsize="17"/>
                    <element
path="/article[1]/body[1]/section[3]/p[2]/collectionlink[16]"
exhaustivity="2" size="18" rsize="18"/>

</file>

```

Figure 2 Rel-assessment 289.xml

## 2.5 The Smart Information Retrieval System

The Smart [9] information retrieval system was developed under the supervision of late Gerard Salton of Cornell University over a thirty year period. It is based on the *Vector Space Model* [2]. We use Smart 13.0 for this research. A description of Smart processing with respect to its use in this work follows.

The raw XML documents are parsed to obtain a set of elements (i.e. paragraphs, sections, articles etc.). Each element is written out in its own separate file. These elements are fed to Smart which treats them as documents. A unique id is assigned to each element fed to Smart, and a file that maps the element-ids to their physical location is generated (the *textloc* file). A *dictionary*, a set of document vectors (*doc.nnn*), and an inverted index (*inv.words*) are also generated. Collection frequency statistics are stored in vector form in a file called *collstats*.

After the elements have been indexed to generate document and query vectors, they are weighted using the *Lnu-ltu* weighting scheme [7] (further described in Chapter 3). Documents weights are computed using the *Lnu* weighting scheme while the queries are weighted using the *ltu* weighting scheme. This weighting scheme also normalizes the term weights based on length of containing document [10]. Retrieval then takes place when the *inner product* is applied to measure the similarity between element and query vectors. Smart then outputs a list of document elements, sorted by similarity score. The output from Smart, which we use as input to our system, guides Flex towards documents that contain potentially relevant content.

## 2.6 Document Length Normalization

Information systems have to deal with documents of varying lengths. Without any form of *document length normalization* [10], a longer document that contains higher occurrences of the query term will have a higher probability of being retrieved by the retrieval system. It is shown in [10] that without considering length and normalization of documents, the retrieval system would produce results that are biased towards longer

documents. This is primarily because they have more terms, as well as terms with higher frequencies.

Also according to [10], the probability of retrieval of a document is inversely proportional to the value of its normalization factor. Thus, we need to increase the normalization factor for longer documents and offset it for shorter ones. For this purpose, we use the *Lnu-ltu* weighting scheme. This weighting scheme provides the necessary normalization that is required to offset the bias towards longer documents.

The *Lnu-ltu* weighting scheme depends on two parameters, namely *slope* and *pivot*, which are empirically determined constants. *Pivot* represents the document length for which the probability of relevance is equal to the probability of retrieval. The normalization factor is pivoted at *pivot* and tilted so that documents on one side of the pivot get larger normalization factor values and those on the other side get smaller values. The amount of tilting required is represented by *slope* and is the harder to determine of the two empirical constants [10]. Figure 3 represents this idea pictorially:

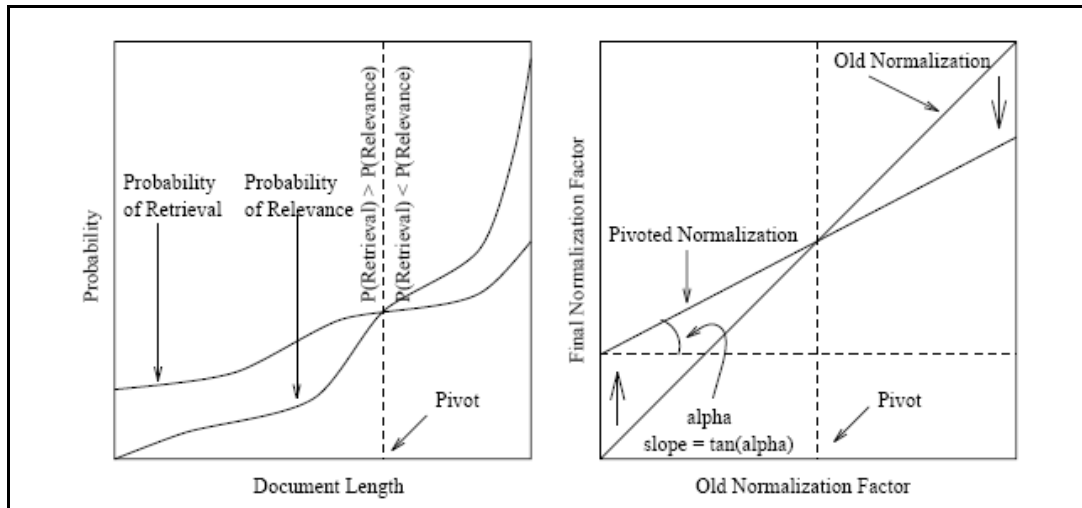


Figure 3 Pivoted document length normalization [10]

The *Lnu* weighting scheme is implemented in Flex. The *nnn* weights (produced by Smart) of the element vectors are converted to *Lnu* weights (applied by Flex) using the formula shown in Figure 4.

$$\frac{\frac{1 + \log(\text{tf})}{1 + \log(\text{average tf})}}{(1 - \text{slope}) + \text{slope} * (\# \text{ unique terms}) / \text{pivot}}$$

where, tf is term frequency (as in the *nmn* vector)  
average tf is the average term frequency of all terms in this vector  
# unique terms is the number of distinct terms in this vector  
slope & pivot are empirically determined constants.

Figure 4 *Lnu* element term weighting formula

Applying the *Lnu* weighting scheme gives us element vectors with term weights that are normalized with respect to the vector length and thereby helps offsets the advantage that longer elements have over shorter ones. Moreover, the computation of *Lnu* term weights for an element vector does not depend on any global statistic, such as inverse document frequency, and can be calculated directly from the *nmn* term weights and the *slope* and *pivot* values (which we determine empirically). The *ltu* weighting scheme, also implemented in Flex, is used to weight query vectors. This is represented mathematically in Figure 5.

$$\frac{(1 + \log(\text{tf})) * \log(N/n_k)}{(1 - \text{slope}) + \text{slope} * (\# \text{ unique terms}) / \text{pivot}}$$

where tf is the term frequency  
N is the collection size  
n<sub>k</sub> is the number of documents that contain this term  
slope & pivot are empirically determined constants  
# unique terms is the number of distinct terms in this vector

Figure 5 *ltu* query term weighting formula [10]

### 3. Retrieval Tasks and Evaluation Metrics

In this chapter we describe the various retrieval tasks that we participate in and the evaluation metrics used at INEX to score the results submitted by participating groups.

#### 3.1 Retrieval Tasks - 2006

UMD participates in the *Ad hoc retrieval track* of INEX [1]. The objective of this task is to retrieve *relevant elements*. INEX regards as relevant elements that both:-

- contain relevant information (the result exhaustively discusses the topic), and
- contain specific information (the result has as little non-relevant information as possible)

The *Ad hoc retrieval track* for the years 2006 is comprised of the following tasks:

- a) **Thorough Task:-** The aim of this task is to return a ranked list of relevant XML elements, in descending order of relevance. The goal is to test a retrieval system's effectiveness in producing a correct ranking. Results for Thorough task can contain overlapping elements.
- b) **Focused Task: -** This task requires a user to return a ranked list of most focused elements that satisfy a user's information need. Moreover, this list should be free of any overlapping elements. This means that for any element present in the result list, no other element along that path in the XML tree of the parent document can be returned.
- c) **Relevant In Context Task: -** The objective of this task is to group together all the relevant elements of a document. To accomplish this, we first identify all the relevant articles during the *fetch phase*. Then we use this ranked list of articles to retrieve all highly correlating elements of each of these articles, and group them together. This is also termed as the *browse phase*. This task is similar to the Fetch and Browse task of INEX 2005. An additional constraint for this task is that the results should be non-overlapping, just like the focused task.
- d) **Best In Context Task:-** For each document that was marked relevant by human assessors, they were also required to indicate a starting point to begin reading that document in order to satisfy the information need in the best manner. This point is

called the Best Entry Point for the document (BEP). The objective of this task was to find these BEPs for articles that were found relevant. Again, the results could include either elements or passages.

### 3.2 Retrieval Tasks - 2007

The Thorough task was dropped by INEX in 2007, but we continued our experimentation for this task to get an estimate of our retrieval system's ability to identify relevant retrievable elements. The other three tasks, namely Focused, Relevant In Context and Best In Context, were continued without any changes. However, INEX 2007 now allows us to return not just well-defined elements in response to a query but also arbitrary *passages* spanning multiple elements, unlike INEX 2006 specifications. A *passage* could span across an element's boundary and need not start or end at the opening and closing tags of an XML element. In other words, a *passage* is simply a piece of relevant text in the document. In our current implementation, we deal with *passages* having well-defined boundaries, i.e., those that start and end at an XML tag (further described in Chapter 4).

### 3.3 Evaluation Metrics

Prior to 2006, when INEX switched to *eXtended CumulatedGain* or xCG based measures [6] for evaluation of content-oriented XML retrieval, *inex-eval* was used as a standard metric. The *inex-eval* metric measures a retrieval system's output as a function of *recall* and *precision*. The definitions of *recall* and *precision* are given in Figure 6:

$$\text{Recall} = \frac{\text{Number relevant documents retrieved}}{\text{Total relevant documents in the collection}}$$
$$\text{Precision} = \frac{\text{Number relevant documents retrieved}}{\text{Total documents retrieved}}$$

Figure 6 recall and precision measures

The *inex-eval* measure relied on the assumed independence of the units of retrieval and a simple model of user interface. The typical retrieval task of such a system, often referred to as flat document retrieval, is to return a ranked list of documents in response to a user's

query. The user presumably reads from the ranked list in a linear fashion until the end of the list is reached, his information need is satisfied, or he gives up.

This view changes in an XML retrieval environment. The *inex-eval* measure did not take into account the following two aspects of XML retrieval, which led to adoption of xCG based metrics for evaluation:

- *near-misses*: Unlike traditional IR, users of an XML IR system also have access to the neighbor (sibling, children or parent) nodes of the elements returned by the system. To simply consider these neighboring elements as non-hits, or *near-misses*, is seen as a rather strict evaluation scenario. This then calls for the need for an evaluation metric that grants partial scores for such near-misses.
- *overlap*: Another aspect of XML retrieval that the evaluation metric must take into account is the retrieval of overlapping elements (e.g., when both a paragraph and its containing section are retrieved). Studies have shown that overlapping and redundant result elements do not add any additional value to users and can lead to frustration. However, *inex-eval* actually ranks IR systems that return overlapping elements higher than those that do not. This also prompted the shift to a measure that would offset the bias towards techniques retrieving redundant content.

The xCG family of metrics considers dependency of XML elements (overlap and near-misses) and is based on the notion of an *ideal recall-base*. An ideal recall-base is the set of best or the ideal answers to a users information need, with all overlapping nodes removed. The ideal recall is constructed from the relevance assessments that have been manually performed by the assessors. Given any two components (marked relevant by the human assessors) that lie on the same relevance path, the component with the higher quantized score is selected. In case the two components' scores are equal, the one higher in the document tree is chosen, i.e., the parent/ascendant.

xCG is defined as a vector of accumulated gain. Given a ranked list of document elements, the cumulated gain at rank  $i$ , denoted  $xCG[i]$ , is computed as the sum.

$$xCG[i] := \sum_{j=1}^i xG[j]$$

where  $xG[j]$  represents the relevance score for  $j^{\text{th}}$  element in the ranked list. For example, the ranking  $xGq = \langle 3,1,0,0,1,3,2,2,0,0 \rangle$  produces the cumulated gain vector of  $xCG = \langle 3,4,4,4,5,8,10,12,12,12 \rangle$ .  $xCG$  value for a retrieval run can then be compared with the cumulated gain vector for the ideal recall-base by plotting both the actual and the ideal gain functions against the rank position. By dividing the  $xCG$  vectors of the retrieval runs by their corresponding ideal  $xCI$  vectors, we obtain the normalized  $xCG$  ( $nxCG$ ) measure:

$$nxCG[i] := \frac{xCG[i]}{xCI[i]}$$

## 4. Our Approach

The processes of all-element and flexible retrieval are represented in Figure 7 and 8.

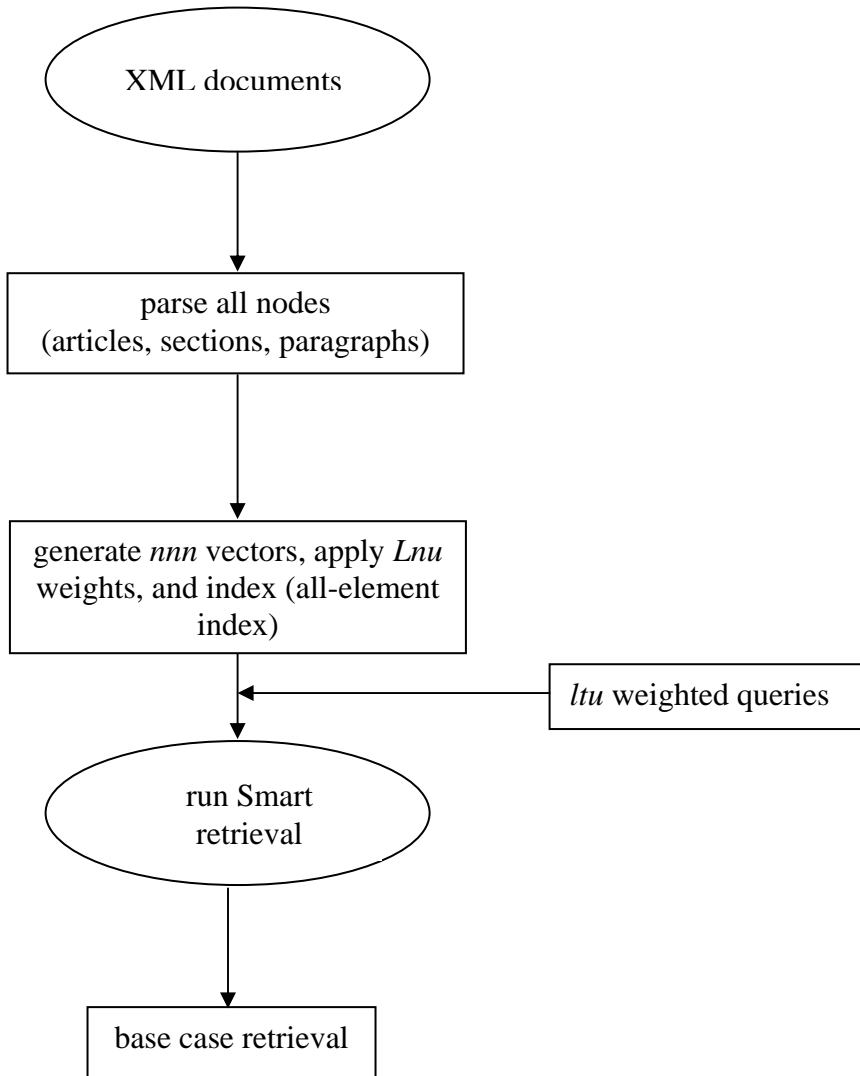


Figure 7 Flow diagram for all-element retrieval

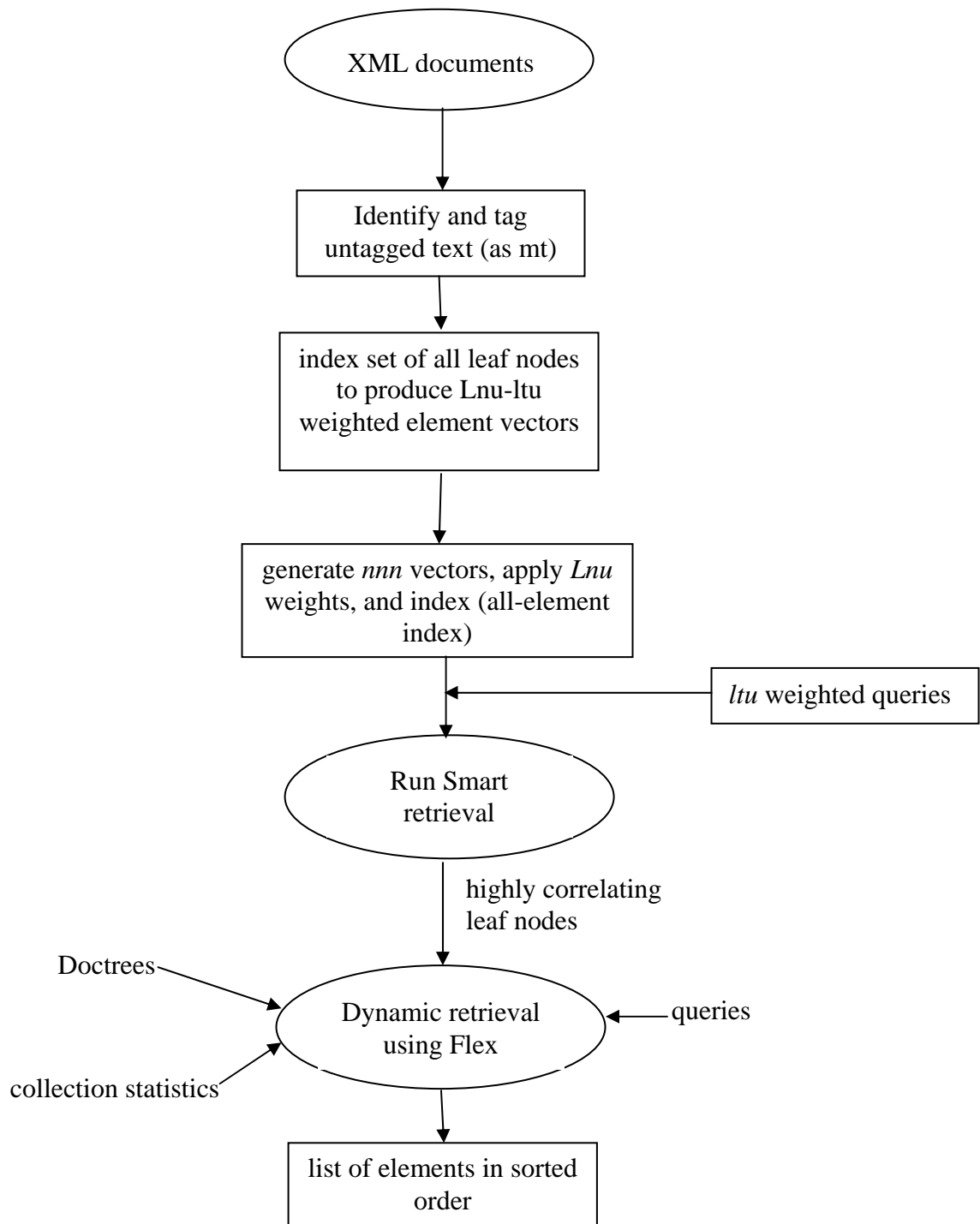


Figure 8 Flow diagram for flexible retrieval

## 4.1 Untagged Text

An important issue lies with the fact that Wikipedia documents are only semi-structured and contain pieces of text that are untagged. Flex performs a vector addition of leaf nodes to build a complete document tree from the bottom up. For this to work, all leaf nodes must be identified before the parent can be produced. We handle this problem by pre-processing the Wikipedia collection. This involves the identification and mark-up of all pieces of untagged text and grouping them as a single element (which we call the *magic text (mt)* element) within a parent.

## 4.2 Parsing

After all untagged text is marked-up, we proceed to parse out XML elements. Two parses are generated: one consisting only of paragraphs and *mts* (*leaf-node parse*), and the other consisting of body, section and paragraph elements for every Wikipedia document (*all-element parse*). (The queries are also parsed to identify title, description and structural constraints). Parsing of XML documents produces new, smaller documents, each consisting of only one element from the original XML document. We also store the XML schema (pre-order traversal of the document tree) of each document as it is parsed in a separate file. This file (the *doctree*) is used later by Flex. A typical *doctree* representation for an XML document is shown in Figure 9.

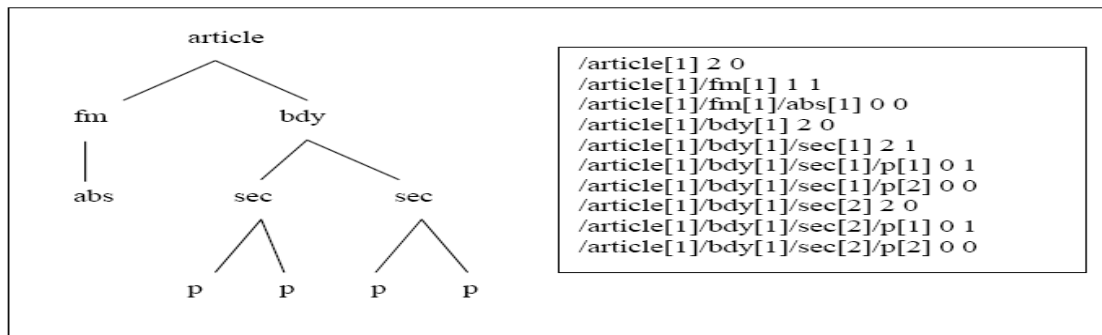


Figure 9 Doctree for an XML document [7]

## 4.3 Smart Retrieval

The all-element parse (consisting of all bodies, sections and paragraphs from the documents) is then indexed and an *Lnu-ltu* retrieval is performed. The all-element index,

which is an overlapping index of every node of each document, is considered to have the best chance of retrieving the most relevant elements. Thus we use this retrieval as a *base case* to which Flex runs are compared.

A second retrieval is done against the *paragraph* index to obtain a list of highly correlating leaf nodes. This set, which serves as input to Flex, gives us an idea of which documents are relevant to the query.

#### 4.4 Flex Retrieval/Dynamic Retrieval

Flex builds complete document tree for each document that contains at least one relevant leaf node in the list of highly correlating terminal nodes identified above. It then reads the *doctrees* generated earlier and loads the schema of each document it will generate in to memory. Inner node of a document is generated by adding the child node vectors. This merging (vector addition) of *nnn* (i.e., term-frequency weighted) vectors is shown in Figure 10.

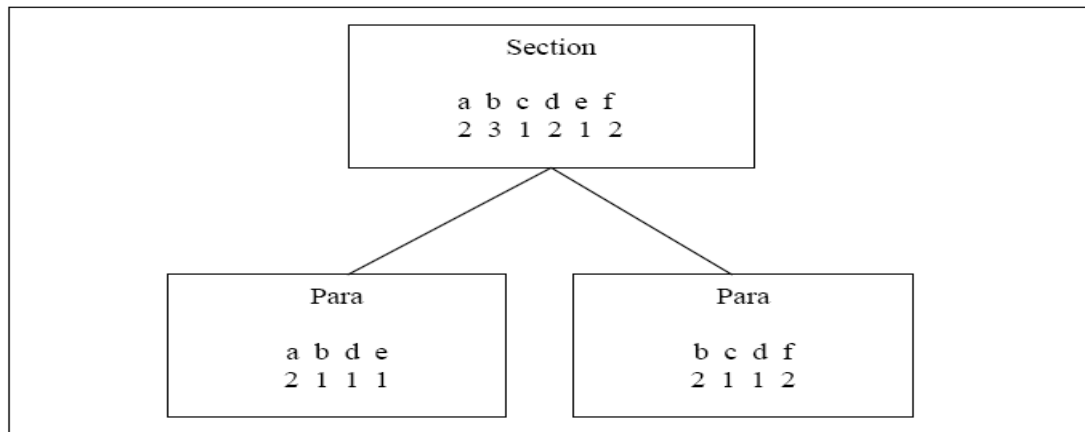


Figure 10 Merging *nnn* vectors to generate inner node vectors [7]

At the end of this step, Flex has generated *nnn*-weighted trees for documents which contain elements that correlate highly with the query. *Lnu* weights are then applied to the element vectors and the queries are weighted using the *ltu* weighting scheme. Finally, an inner product (vector dot product) of these elements with the query vector is computed, producing a ranked list of elements based on similarity score.

## 4.5 The Weighting Scheme

Traditional weighting schemes produce a bias towards longer elements with more terms and terms with higher term frequency. In our work, we address this bias by using the *Lnu-ltu* weighting scheme [10] discussed in Chapter 3. Application of these weights requires tuning the *slope* and *pivot* values for the collection. The *pivot*, which represents the average document length, is easily determined. *Slope* is determined by carrying out numerous retrievals at different values of slope, and evaluating the results to find the value which produces the best results in terms of the appropriate metric.

## 5. Experiments and Results

This chapter describes the two experiments performed on the 2006 Wikipedia collection as well as the results obtained when these experiments are applied to each of the Ad hoc subtasks.

### 5.1 Experiments

In this section, we consider particular aspects of retrieval related to (1) terminal node expansion and (2) structural constraints on queries.

#### Terminal Node Expansion

Most Wikipedia tags contain fewer than 5 words on average. This translates to many small elements with no or little meaning from a user's point of view. A design issues we had to address was to decide which elements to choose as our leaf or terminal nodes. In particular, we wanted to select the smallest meaningful elements as our leaf nodes and at the same time disregard those that were either too small to be returned or lacked meaning. With this in mind, we had previously chosen a set of meaningful terminal nodes (which we refer to as paragraphs). A list of these nodes is given in Table 5.

Table 5 Terminal Nodes

<b>Terminal Nodes</b>
paragraph
numberlist
ordered list
unordered list
figure
definition list
item
table
normallist

However, all of these elements which we refer to generically as paragraphs, may contain smaller elements (e.g., emph, cell, etc.) that, while meaningless in themselves, may nevertheless improve the results if included in an INEX evaluation. To investigate this

issue, we include a *post-processing* phase which allows us to expand our defined set of terminal nodes by including all the lower level nodes contained within them.

We made a decision earlier in our system design phase to discard a number of low level tags, because their average length is 2 words (which we considered too little information to be meaningful) and because adding these tags resulted in a huge index due to their sheer number in the collection. But some of these tags (e.g., *collectionlink*, *unknownlink*, *emph3*, *cell*, *caption*, etc.) appear in the corresponding relevance assessments against which our results are judged. Since we do not include them in our index, our system can not retrieve any of these elements. To get as close to the elements listed in relevance-assessments as possible, we post-process our results to expand each leaf node with all the lower level elements (*terminal node expansion*) contained within it. We assign these elements the same similarity score as that of their containing parent (the leaf node). Figure 11 below shows a sample original retrieval output of our system and the same retrieval after post-processing it by terminal node expansion.

1 231/article[1]/body[1]/section[1] 44.2974	1 231/article[1]/body[1]/section[1] 44.2974
1 288/article[1]/body[1]/section[3] 40.7043	1 288/article[1]/body[1]/section[3] 40.7043
1 49/article[1]/body[1]/p[5] 38.9533	1 49/article[1]/body[1]/p[5] 38.9533
1 150/article[1]/body[1]/section[6] 38.2962	1 49/article[1]/body[1]/p[5]/collectionlink[1] 38.9533
1 734/article[1]/body[1]/section[1]/p[4] 37.8511	1 49/article[1]/body[1]/p[5]/collectionlink[2] 38.9533
1 288/article[1]/body[1]/section[3]/p[2] 37.7817	1 49/article[1]/body[1]/p[5]/collectionlink[3] 38.9533
	1 150/article[1]/body[1]/section[6] 38.2962
	1 734/article[1]/body[1]/section[1]/p[4] 37.8511
	1 734/article[1]/body[1]/section[1]/p[4]/unknownlink[1] 37.8511
	1 734/article[1]/body[1]/section[1]/p[4]/collectionlink[1] 37.8511
	1 288/article[1]/body[1]/section[3]/p[2] 37.7817
	1 288/article[1]/body[1]/section[3]/p[2]/caption[1] 37.7817
	1 288/article[1]/body[1]/section[3]/p[2]/emph3[1] 37.7817

Figure 11 Original retrieval and post-processed retrieval

### Structural Constraints on Queries (CAS)

This set of experiments considers structural constraints imposed by queries on the type of elements desired as output. Structural constraints impose one or more conditions on the

query, requiring that a specific element type be returned. These structural constraints apply only to the CAS query set. Consider table 6 and 7 below.

Table 6 INEX 2006 queries and structural constraints

	<b>Topics-2006 125 topics</b>	
<b>structural constraint</b>	<b>queries-ids with these constraints</b>	<b>Total no of queries</b>
all	289 296 302 302 305 309 319 320 323 328 330 332 335 336 339 342 344 348 350 351 352 352 354 363 373 374 380 382 383 385 387 393394 395 397 404 406 408	38
body	290 301 306 312 340 341 346 356 365 378 379 388 325 343 349	15
p	300 314 315 338 364 366 375 384 386 403 407 409 410 411 412	15
section	293 294 295 297 298 299 304 307 308 310 311 313 316 317 318 321 322 324 326 327 329 333 334 337 345 355 357 358 359 360 361 362 367 368 369 370 372 376 377 381 389 398 400 401 402 413 390 391 392	49
caption	396	1
figure	291 292 331 347	4
table	399	1
unknownlink	-	0
collectionlink	371 405	2
emph3	-	0
cell	-	0
item	-	0
normallist	-	0
numberlist	-	0
row	-	0

Table 7 INEX 2007 queries and structural constraints.

	<b>Topics-2007 127 topics</b>	
<b>elements in rel-assessments</b>	<b>queries-ids with these constraints</b>	<b>Total no of queries</b>
all	417 425 433 441 449 450 451 452 453 455 456 457 458 460 462 465 467 468 469 471 473 475 484 492 493 498 499 502 503 504 505 506 507 509 510 512 513 514 516 517 518 519 520 521 522 523 524	47
body	500 419 424 426 427 429 430 432 435 446 447 478 479 480 481 482 483 485 486 488 490 495 496 501 515 445 497	27
p	418 422 444 464 470 474 532	7
section	440 442 443 454 463 476 491 420 421 423 428 431 434 436 437 438 439 448 459 466 477 487 489 508 511 461	26
caption	539	1
figure	472 525 528 529 530 531 535 540	8
image	526 527 533 534 536 537 538 541 542 543	10
title	494	1
table	-	0
unknownlink	-	0
collectionlink	-	0
emph3	-	0
cell	-	0
item	-	0
normallist	-	0
numberlist	-	0
row	-	0

These tables show that there are in fact some low level tags omitted in our processing which specific CAS queries must return. Theoretically, imposing structural constraints on the result set should give us better scores on CAS evaluations.

With these two factors (terminal node expansion and imposition of structural constraints on queries) in mind, we performed a set of experiments as follows.

For CO queries

- without terminal node expansion (base-case) vs. with terminal node expansion.  
(base case vs. post-processing of results)

For CAS queries

- without terminal node expansion (base-case) vs. with terminal node expansion.  
(base case with structural constraints applied vs. the same case with post-processing of results)

These experiments were applied to each of the four basic Ad hoc subtasks: Thorough, Focused, Relevant In Context and Best In Context. All experiments using terminal node expansion were performed on the output (terminal node element set) produced by Flex, for different values of  $n$  (number of input paragraphs) and  $N$  (rank). Slope and pivot values of 0.14 and 120 respectively were used for the base case retrieval. Slope and pivot values of 0.12 and 18, respectively, were used for the paragraph retrieval which determines the input to Flex. The results are shown in the following section.

## 5.2 Results

Experiments for the following cases are performed and reported below. For each result table, the first column represents  $n$  (number of leaf nodes input to Flex), and the first row represents the value of the metric at rank  $N$ . For the Thorough task, for example, the @20 column indicates the MAep value after 20 elements have been retrieved. The best result at each rank is shown in bold. The first row of values represents the base case. Smart retrieval carried out against the all-element index, whereas the remaining rows are values obtained by evaluating Flex results.

## Thorough Subtask

For CO queries:

- *Thorough CO with no terminal node expansion* – Original results from Flex retrievals for different values of  $n$  and  $N$  are compared against the base case retrieval performed against the all-element index (Table 8.1) [Base case]
- *Thorough CO with terminal node expansion* – For this case, all the results undergo a phase of terminal node expansion before being evaluated (Table 8.2)

For CAS queries:

- *Thorough CAS with no terminal node expansion* – In this case, before evaluating the results, we apply a constraint filter that processes the results to comply with the structural constraints listed in Table 6 for 2006 queries (Table 8.3) [Base case]
- *Thorough CAS with terminal node expansion* – We expand the terminal nodes and then apply the constraints filter before evaluating the results in this case (Table 8.4)

Table 8.1 Thorough CO with no terminal node expansion (MAep)

$n$	@10	@20	@50	@100	@500	@1500
base case	0.0045	0.0065	0.0098	0.0126	0.0171	0.0188
1000	0.0045	0.0065	0.0098	0.0126	0.0171	0.0188
500	0.0045	0.0065	0.0098	0.0126	0.0172	0.0196
250	0.0045	0.0065	0.0098	0.0125	0.0175	0.0209
100	<b>0.0046</b>	0.0065	0.0098	0.0125	0.019	<b>0.0219</b>
50	0.0045	0.0064	0.0097	0.0129	<b>0.0203</b>	0.0212
25	0.0045	0.0063	0.0100	0.0138	0.0192	0.0192
10	0.0045	0.0067	0.0111	<b>0.014</b>	0.0158	0.0158
5	0.0045	<b>0.0072</b>	<b>0.0113</b>	0.0129	0.0133	0.0133
1	0.0045	0.0061	0.0075	0.0078	0.0078	0.0078

Table 8.2 Thorough CO with terminal node expansion (MAep)

$n$	@10	@20	@50	@100	@500	@1500
base case	0.0055	0.0088	0.0148	0.0202	0.0351	0.0414
1000	0.0055	0.0088	0.0148	0.0202	0.0351	0.0414
500	0.0055	0.0088	0.0148	0.0203	0.0351	0.04151
250	0.0055	0.0088	0.0148	0.0203	0.035	0.0429
100	0.0055	0.0088	0.0148	0.0202	0.036	<b>0.0466</b>
50	<b>0.0055</b>	0.0088	0.0147	0.0202	<b>0.0398</b>	0.0474
25	0.0054	0.0088	0.0148	0.0209	0.0399	0.0439
10	0.0054	0.0088	0.0162	<b>0.0238</b>	0.0364	0.0372
5	0.0054	<b>0.0093</b>	<b>0.0167</b>	0.0236	0.0312	0.0312

Table 8.3 Thorough CAS with no terminal node expansion (MAep)

$n$	@10	@20	@50	@100	@500	@1500
base case	0.0033	0.0046	0.0064	0.0077	0.0098	0.0101
1000	0.0035	0.0047	0.0064	0.0077	0.0099	0.0101
500	0.0036	0.0047	0.0065	0.0077	0.0098	0.0106
250	0.0035	0.0047	0.0065	0.0077	0.0099	<b>0.0108</b>
100	0.0036	0.0047	0.0065	0.0078	0.0101	0.0106
50	<b>0.0036</b>	<b>0.0048</b>	0.0066	<b>0.0079</b>	<b>0.0101</b>	0.0102
25	0.0035	0.0047	0.0065	0.0078	0.0090	0.0090
10	0.0034	0.0045	0.0063	0.0072	0.0075	0.0075
5	0.0032	0.0044	<b>0.0097</b>	0.0061	0.0061	0.0061

Table 8.4 Thorough CAS with terminal node expansion (MAep)

$N$	@10	@20	@50	@100	@500	@1500
base case	0.0043	0.0063	0.0091	0.0111	0.0161	0.0187
1000	0.0043	0.0063	0.0091	0.0112	0.0164	0.0188
500	0.0043	0.0063	0.0091	0.0112	0.0165	0.0190
250	0.0043	0.0063	0.0091	0.0112	0.0166	0.0020
100	0.0043	0.0063	0.0091	0.0113	0.0168	<b>0.0205</b>
50	0.0044	<b>0.0064</b>	0.0092	0.0114	<b>0.0174</b>	0.0202
25	<b>0.0044</b>	0.0063	0.0092	0.0113	0.016	0.0192
10	0.0042	0.0060	0.0089	<b>0.0118</b>	0.0141	0.0168
5	0.0040	0.0058	<b>0.0087</b>	0.0111	0.0139	0.0139

### Analysis of Results

We see a significant boost over our original CO Thorough results after they were post-processed. The MAep value of 0.0466 for  $n=100$  and  $N=1500$  for *CO with terminal node expansion* case places us at rank 1 in the 2006 INEX rankings compared to the corresponding value for our original unprocessed results that places us at 25. We also notice that we get better results at lower values of  $n$ . In fact, all the high scores for the Thorough task occur in the  $n=5$  to 100 range. The post-processed CAS results were marginally lower than the original results.

### Focused sub-task

We perform the following experiments for the Focused subtask.

For CO queries:

- *Focused (overlap OFF) CO with no terminal node expansion (Table 9.1)[Base Case]*
- *Focused (overlap OFF) CO with terminal node expansion (Table 9.2)*
- *Focused (overlap ON) CO with no terminal node expansion (Table 9.3)[Base Case]*
- *Focused (overlap ON) CO with terminal node expansion (Table 9.4)*

For CAS queries:

- *Focused (overlap OFF) CAS with no terminal node expansion (Table 9.5)[Base Case]*
- *Focused (overlap OFF) CAS with terminal node expansion (Table 9.6)*
- *Focused (overlap ON) CAS with no terminal node expansion (Table 9.7)[Base Case]*
- *Focused (overlap ON) CAS with terminal node expansion (Table 9.8)*

Table 9. 1 Focused (overlap OFF) CO with no terminal node expansion (nxCG)

<i>n</i>	@5	@10	@25	@50
base case	0.3625	0.3134	0.2538	0.2042
1000	0.3645	0.3122	0.2526	0.2036
500	0.3605	0.3098	0.2487	0.2005
250	0.3595	0.3124	0.2481	0.1993
100	0.3595	0.3100	0.2479	0.2019
50	0.3595	0.3108	0.2529	0.2221
25	0.3619	0.3159	0.2791	<b>0.2518</b>
10	0.3763	0.3477	0.3104	0.2508
5	<b>0.3993</b>	<b>0.3684</b>	<b>0.3135</b>	0.2111
1	0.3321	0.261	0.1686	0.0992

Table 9. 2 Focused (overlap OFF) CO with terminal node expansion (nxCG)

<i>n</i>	@5	@10	@25	@50
base case	0.4358	0.4086	0.3428	0.3030
1000	0.4341	0.4094	0.3428	0.3030
500	0.4296	0.4090	0.3425	0.3024
250	0.4241	0.4040	0.3388	0.3001
100	0.4190	0.4023	0.3366	0.2981
50	0.4190	0.4023	0.3372	0.2991
25	0.4190	0.4041	0.3410	0.3112
10	<b>0.4391</b>	0.4103	0.3801	0.3333
5	0.4361	<b>0.4254</b>	<b>0.4014</b>	<b>0.3488</b>

Table 9. 3 Focused (overlap ON) CO with no node expansion (nxCg)

<i>n</i>	@5	@10	@25	@50
allele	0.3241	0.2787	0.2220	0.1815
1000	0.3261	0.2766	0.2207	0.1811
500	0.3221	0.2745	0.2169	0.1776
250	0.3211	0.2748	0.2159	0.1764
100	0.3211	0.2748	0.2157	0.1779
50	0.3211	0.2756	0.2197	<b>0.1854</b>
25	0.3234	0.2789	<b>0.2307</b>	0.1846
10	<b>0.3333</b>	<b>0.2852</b>	0.2105	0.1542
5	0.3303	0.2601	0.1894	0.1118
1	0.2249	0.1505	0.0843	0.0467

Table 9. 4 Focused (overlap ON) CO with terminal node processing (nxCg)

<i>n</i>	@5	@10	@25	@50
allele	0.2309	0.1596	0.1223	0.1127
1000	<b>0.2309</b>	<b>0.1598</b>	<b>0.1238</b>	0.1129
500	0.2247	0.1594	0.1221	0.1121
250	0.2175	0.1535	0.1166	<b>0.1176</b>
100	0.2141	0.1509	0.1137	0.1040
50	0.2123	0.1509	0.1133	0.1045
25	0.2141	0.1518	0.1170	0.1090
10	0.2100	0.1501	0.1170	0.0099
5	0.2099	0.1434	0.1181	0.0899

#### Analysis of Results (CO Focused)

As noticed in the Thorough case, post-processing gives us significant improvement for the CO Focused (overlap off) task too. Our overall rank based on INEX 2006 rankings, improves from 5 to 2 for nxCg@5, 6 to 2 for nxCg@10, 8 so 1 for nxCg@25 and 9 to 1 for nxCg@50. The results in the CO Focused (overlap ON) case dropped significantly after they are *post-processed*. This is attributed to the fact that the evaluation of the

Focused ON task awards no partial credit for *near-misses*. And by expanding a paragraph into all of its component elements, we may still only be nodes that are close to a relevant node but not the node itself.

Table 9. 5 Focused (overlap OFF) CAS with no terminal node expansion (nxCg)

<i>n</i>	@5	@10	@25	@50
allele	0.3821	0.3017	0.2351	0.1782
1000	<b>0.3937</b>	0.3187	0.2353	0.1785
500	0.382	0.3197	0.2359	0.1784
250	0.3792	0.3198	<b>0.2359</b>	0.1790
100	0.3788	<b>0.3228</b>	0.2353	0.1783
50	0.3818	0.3169	0.2360	<b>0.1885</b>
25	0.3711	0.2854	0.2379	0.1896
10	0.3608	0.2857	0.2212	0.1626
5	0.3397	0.2895	0.2112	0.1269

Table 9. 6 Focused (overlap OFF) CAS with terminal node expansion (nxCg)

<i>n</i>	@5	@10	@25	@50
allele	0.4315	0.3826	0.2786	0.2219
1000	<b>0.4315</b>	0.3807	0.2766	0.2199
500	0.4298	0.3826	0.2786	0.2219
250	0.4279	0.3814	0.2800	0.2230
100	0.4266	<b>0.3844</b>	<b>0.2788</b>	0.2221
50	0.4297	0.3793	0.2787	<b>0.2249</b>
25	0.4173	0.3670	0.2769	0.2242
10	0.3803	0.3510	0.2680	0.2212
5	0.3775	0.3351	0.2670	0.1988

Table 9. 7 Focused (overlap ON) CAS with no terminal node expansion (nxCG)

<i>n</i>	@5	@10	@25	@50
allele	0.3608	0.2903	0.2091	0.1601
1000	<b>0.361</b>	0.2893	0.2095	0.1605
500	0.3592	0.2903	0.2100	0.1602
250	0.3566	0.2907	<b>0.2100</b>	<b>0.1605</b>
100	0.3561	<b>0.2938</b>	0.2083	0.1574
50	0.3591	0.2878	0.2054	0.1568
25	0.3502	0.2790	0.1996	0.1447
10	0.3301	0.2461	0.1523	0.0976
5	0.3083	0.2226	0.1328	0.0769

Table 9. 8 Focused (overlap ON) CAS with terminal node expansion (nxCG)

N	@5	@10	@25	@50
allele	0.2968	0.2328	0.1628	0.1301
1000	<b>0.2968</b>	0.2337	0.1634	0.1319
500	0.2951	0.2356	0.1650	<b>0.1338</b>
250	0.2931	0.2335	<b>0.1654</b>	0.1340
100	0.2919	<b>0.2366</b>	0.1631	0.1299
50	0.2950	0.2315	0.1590	0.1293
25	0.2843	0.2217	0.1571	0.1229
10	0.2841	0.2121	0.1571	0.1201
5	0.2391	0.1750	0.1146	0.0713

Analysis of Results (CAS Focused)

As seen in the CO Focused case, the results for CAS Focused runs were significantly better for the overlap off case. However, we see a drop in results in the overlap on case due to the penalty for near misses.

## Relevant In Context (RIC) sub-task

Experiments for the following case are performed and reported below.

For CO queries:

- *RIC CO with no terminal node expansion (Table 10.1) [Base Case]*
- *RIC CO with terminal node expansion (Table 10.2)*

For CAS queries:

- *RIC CAS with no terminal node expansion (Table 10.3) [Base Case]*
- *RIC CAS with terminal node expansion (Table 10.4)*

Table 10. 1 Relevant In Context CO with no terminal node expansion (gP)

$n$	gP[5]	gP[10]	gP[25]	gP[50]	MAgp
base case	0.0947	0.0718	0.0538	0.0396	0.0337
1000	0.0948	0.0706	0.0545	0.0400	0.0343
500	0.1241	0.0924	0.0717	0.0525	0.0445
250	0.1550	0.1116	0.0849	0.0608	<b>0.0519</b>
100	0.1927	0.1384	0.1086	<b>0.0708</b>	0.0563
50	0.2208	0.1659	0.1131	0.0663	0.0549
25	0.2632	0.2137	<b>0.1216</b>	0.0608	0.0544
10	<b>0.3279</b>	<b>0.2241</b>	0.0896	0.0448	0.0453
5	0.2721	0.1360	0.0544	0.0272	0.0286

Table 10. 2 Relevant In Context CO with terminal node expansion (gP)

$n$	gP[5]	gP[10]	gP[25]	gP[50]	MAgp
allele	0.0642	0.0626	0.0453	0.0305	0.0285
1000	0.0732	0.0548	0.0446	0.0335	0.0286
500	0.0755	0.0575	0.0465	0.0355	0.0300
250	0.0825	0.0640	0.0519	0.0396	0.0320
100	0.1222	0.0901	0.0658	<b>0.0442</b>	<b>0.0370</b>
50	0.1403	0.1042	0.0678	0.0385	0.0338
25	0.1638	0.1276	<b>0.0683</b>	0.0341	0.0330
10	<b>0.2082</b>	<b>0.1338</b>	0.0535	0.0267	0.0285
5	0.1781	0.0892	0.0357	0.0178	0.0190

Table 10. 3 Relevant In Context CAS with no terminal node expansion (gP)

<i>n</i>	gP[5]	gP[10]	gP[25]	gP[50]	MAgp
allele	0.1025	0.0700	0.0511	0.0375	0.0306
1000	0.1036	0.0752	0.0551	0.0377	0.0306
500	0.1300	0.0968	0.0737	0.0504	0.0400
250	0.1602	0.1192	0.0874	0.0572	0.0465
100	0.1904	0.1426	<b>0.1016</b>	<b>0.0655</b>	0.0481
50	0.2093	0.1611	0.1004	0.0551	0.0447
25	0.2423	<b>0.1951</b>	0.0994	0.0497	0.0409
10	<b>0.2721</b>	0.1589	0.0635	0.0317	<b>0.0523</b>
5	0.2116	0.1058	0.0423	0.0211	0.0396

Table 10. 4 Relevant In Context CAS with terminal node expansion (gP)

<i>n</i>	gP[5]	gP[10]	gP[25]	gP[50]	MAgp
base case	0.0880	0.0871	0.0640	0.0485	0.0411
1000	0.0970	0.0884	0.0644	0.0485	0.0412
500	0.1066	0.0963	0.0702	0.0526	0.0441
250	0.1065	0.0945	0.0689	0.0517	0.0427
100	0.1607	0.1249	<b>0.0973</b>	<b>0.0654</b>	<b>0.0453</b>
50	0.1866	0.1467	0.0935	0.0513	0.0390
25	0.2146	<b>0.1808</b>	0.0894	0.0447	0.0356
10	<b>0.2372</b>	0.1395	0.0558	0.0279	0.0444
5	0.1887	0.0943	0.0377	0.0188	0.0331

### Analysis of Results

Relevant In Context is a new task that was added to the Ad hoc retrieval track in 2006. Based on the experiments above, we noticed that the results with terminal node expansion for CO and CAS queries are slightly lower than those with no post-processing.

## Best In Context (BIC) sub-task

We repeated the same set of experiments for this sub-task too.

For CO queries:

- *BIC CO with no terminal node expansion (Table 11.1)* [Base Case]
- *BIC CO with terminal node expansion (Table 11.2)*

For CAS queries:

- *BIC CAS with no terminal node expansion (Table 11.3)* [Base Case]
- *BIC CAS with terminal node expansion (Table 11.4)*

Table 11. 1 BEP CO with no terminal node expansion (BEPD)

A ->	0.01	0.1	1	10	100
	0.0321	0.0508	0.0781	0.0114	0.015
1000	0.0323	0.0514	0.0782	0.1148	0.1508
500	0.0388	0.0611	<b>0.0928</b>	0.1341	0.1742
250	<b>0.0394</b>	<b>0.0613</b>	0.0925	<b>0.1343</b>	<b>0.1758</b>
100	0.0355	0.0566	0.0861	0.1262	0.1661
50	0.0288	0.0475	0.0738	0.1102	0.1465
25	0.0268	0.0441	0.0682	0.0979	0.1277
10	0.0141	0.0276	0.046	0.0693	0.0944
5	0.0056	0.0142	0.0282	0.0446	0.0601

Table 11. 2 BEP CO with terminal node expansion (BEPD)

A ->	0.01	0.1	1	10	100
	0.0373	0.0542	0.0900	0.01307	0.01713
1000	0.0376	0.0596	0.0901	0.1312	0.1714
500	<b>0.0394</b>	<b>0.0615</b>	<b>0.0926</b>	0.1335	0.1729
250	0.0393	0.0611	0.0922	<b>0.1339</b>	<b>0.1754</b>
100	0.0357	0.0566	0.8612	0.1262	0.1661
50	0.0288	0.0475	0.0738	0.1102	0.1465
25	0.0268	0.0441	0.0682	0.0979	0.1277
10	0.0141	0.0276	0.0460	0.0693	0.0944
5	0.0056	0.0142	0.0282	0.0446	0.6017

Analysis of Results (CO Best In Context task)

For the Best In Context task, the results for CO queries, with and without terminal node processing, are comparable. We noticed that results with post-processing are higher at A=0.1 and A=10 as compared to those without any terminal node expansion.

Table 11. 3 BEP CAS with no terminal node expansion (BEPD)

A ->	0.01	0.1	1	10	100
	0.0211	0.0376	0.0608	0.0902	0.0903
1000	0.0219	0.0383	0.0611	0.0908	0.0908
500	<b>0.0307</b>	<b>0.0445</b>	<b>0.0697</b>	0.1077	<b>0.1081</b>
250	0.0286	0.0438	0.0695	<b>0.1071</b>	0.1075
100	0.0249	0.0432	0.0694	0.1070	0.1070
50	0.0192	0.0347	0.0582	0.0913	0.0913
25	0.0192	0.0326	0.0541	0.0817	0.0817
10	0.0146	0.0313	0.0490	0.0745	0.0745
5	0.0121	0.0287	0.0365	0.0612	0.0615

Table 11. 4 BEP CAS with terminal node expansion (BEPD)

A ->	0.01	0.1	1	10	100
	0.0281	0.0467	0.0871	0.1531	0.2214
1000	0.0296	0.0471	0.0873	0.1536	0.2216
500	<b>0.0305</b>	<b>0.0478</b>	<b>0.0894</b>	<b>0.1591</b>	<b>0.2316</b>
250	0.0274	0.0426	0.0806	0.1446	0.2133
100	0.0232	0.0347	0.0580	0.1002	0.1449
50	0.0212	0.0308	0.0495	0.0856	0.1245
25	0.0225	0.0317	0.0490	0.0791	0.1093
10	0.0085	0.0203	0.0379	0.0656	0.1025
5	0.0054	0.0141	0.0271	0.0454	0.0714

Analysis of Results (CAS Best In Context task)

We got significant improvements for post-processed CAS results for the Best In Context task. The BEPD distance values are higher at all A values except at A=0.01 in comparison to those for unprocessed CAS results.

## 6. Conclusions and Future Work

We have successfully extended *dynamic retrieval* to the Wikipedia collection. We made a significant improvement in the performance of our system. This is due to the post-processing scheme adopted in this work. It enables us to retrieve smaller level elements, too small and numerous to be included in our index and yet seen and marked relevant by human assessors.

As shown in Chapter 5, this post-processing of the results obtained from Flex outperforms the original Thorough and Focused overlap OFF runs for both CO and CAS queries. Post-processing of the Focused ON runs produces results comparable to those produced by the original, base case runs. We also get improvement in the expanded Best Entry Point retrievals for CAS queries. Moreover, these post-processed Flex runs also out-perform the base case, all-element retrieval produced by Smart.

This thesis presents an extension of our approach to dynamic retrieval for the Wikipedia collection. It requires neither indexing every document element in the collection nor the use of multiple indices. We use the same techniques to submit our results for the 2007 query set.

With respect to future work: The INEX 2007 specification now allows us to return passages in addition to well defined XML elements in our results. Our current implementation can identify passages that span an XML boundary. A relevance assessor can, however, mark arbitrary pieces of text as relevant passages. Extending our implementation to identify and retrieve arbitrary passages requires a new approach and new methods. The post-processing scheme used assigns an equal score to every low level node as its containing paragraph. An evaluation method to distinguish one from the other could help us improve our results for the Focused ON task.

## References

- [1] Clarke, L.A; Kamps, J; lalmas, M. *INEX 2007 Retrieval Task and Result Submission Specification*, [http://inex.is.informatik.uni-duisburg.de/2007/Ad hoc.html](http://inex.is.informatik.uni-duisburg.de/2007/Ad%20hoc.html).
- [2] Fox, E. *Extending the Boolean and Vector Space Model of Information Retrieval with P-norm Queries and Multiple Concept Types*, Ph.D. Dissertation, Department of Computer Science, Cornell University, 1983.
- [3] Ganpatibhotla, M. *Query Processing in a Flexible Retrieval Environment*, MS Thesis, University of Minnesota Duluth, 2006.
- [4] <http://en.wikipedia.org/wiki/Wikipedia>
- [5] *INEX 2006 Relevance Assessment Guide*, [http://inex.is.informatik.uni-duisburg.de/2006/inex06/Adhocprotected/downloads/Relevance\\_Assessment2006.pdf](http://inex.is.informatik.uni-duisburg.de/2006/inex06/Adhocprotected/downloads/Relevance_Assessment2006.pdf).
- [6] Kazai, G; Lalmas, M. *eXtended Cumulated Gain Measures for the Evaluation of Content-Oriented XML Retrieval*, INEX 2006 Preproceedings.
- [7] Khanna, S. *Design and Implementation of a Flexible Retrieval System*, MS Thesis, University of Minnesota Duluth, 2005.
- [8] Malik, Saadia; Trotman, Andrew; lalmas, Mounia; Fuhr, Norbert. *Overview of INEX 2006*, INEX 2007 Workshop Preproceedings.
- [9] Salton, G., editor, *The SMART Retrieval System - Experiments in Automatic Document Retrieval*, Prentice-Hall, Eaglewood Cliffs, NJ, 1971
- [10] Singhal, A; Buckley, C; Mitra, M. *Pivoted Document Length Normalization*, *Proceedings of the 19<sup>th</sup> Annual International Conference on Research in Information Retrieval*, #.P. 19-21.
- [11] Trotman, Andrew; Larsen, Birger; et al. *INEX 2007 Guidelines for Topic Development*, INEX 2007 Workshop Preproceedings