

Improving Results for the INEX Thorough Task

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

Abhijeet P. Mahule

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

August, 2010

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

Acknowledgements

I would like to take this opportunity to express my gratitude to people who have helped in a direct or indirect way towards completion of this thesis.

Firstly, I would like to thank my advisor Dr. Carolyn Crouch for being a constant source of support. This thesis would not have been possible without her guidance.

I would like to thank the CS faculty at UMD including Dr. Doug Dunham, Dr. Ted Pedersen, Dr. Chris Prince, Dr. Hudson Turner and Dr. Pete Willemsen who imparted wonderful Computer Science knowledge during my MS. A special note of thanks to my Math professor Dr. John Greene whose courses were one of the best I have taken. I am also thankful to Lori Lucia of Computer Science Department for always being there when her help was needed.

I would like to thank my friend Dinesh Bhirud for giving me insight into the information retrieval work done in the past at UMD. I would also like to thank my co-workers Sandeep Vadlamudi and Ramakrishna Cherukuri for their support.

I am thankful to my wife for her unconditional support in all my endeavors. I am also grateful to all my friends who are an integral part of my life.

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 the widespread use of the web and markup languages like XML (Extensible Markup Language) for representing documents, the idea of retrieving at a more granular (element) level evolved. And with the continuous growth in XML information repositories, the focus is now on developing effective element retrieval strategies. Our element retrieval strategy, called Flex (*flexible retrieval*), is designed to work with a semi-structured document collection of XML documents such as the INEX 2009 collection.

In this thesis, we focus on improving the results for the INEX Thorough task by combining article retrieval with Flex for element generation and retrieval. Experiments to determine the best value of slope and pivot are necessary to enable the use of *Lnu-ltu* weighting by Flex. The results of our experiments to produce improved results for the INEX 2009 Thorough task are described and reported.

TABLE OF CONTENTS

List of Tables.....	iv
List of Figures.....	v
1. Introduction.....	1
2. Overview.....	3
2.1 INEX.....	3
2.2 Document Collection.....	4
2.3 Topic or Query Collection.....	4
2.4 Relevance Assessment.....	5
2.5 INEX Retrieval Tasks.....	7
2.6 Evaluation Metric.....	8
2.7 Retrieval Model.....	13
3. Element Retrieval.....	16
3.1 Cleaning, Parsing and Indexing.....	17
3.2 Smart Retrieval.....	22
3.3 Flexible Retrieval (Flex).....	22
3.4 Conversion to Specific Task Format.....	28
3.5 Result Evaluation.....	28
4. Experiments, Results, Analysis.....	32
4.1 All Element Retrieval.....	32
4.2 Thorough Task.....	32
5. Conclusions and Future Work.....	38
References.....	39

List of Tables

Table 1. Details of a Topic.....	5
Table 2. Tags to Keep.....	20
Table 3. Tags to Index.....	21
Table 4. Thorough Task Evaluation for Slope Value = 0.10, Pivot=44...34	
Table 5. Thorough Task Evaluation for Slope Value = 0.11, Pivot=44...35	
Table 6. Thorough Task Evaluation for Slope Value = 0.12, Pivot=44...35	
Table 7. Top-ranked Participants in Thorough Task – 2009.....	37

List of figures

Figure 1: Structure of a Sample Wikipedia Document.....	5
Figure 2: Sample Query.....	6
Figure 3. Formula for Precision at Rank r.....	9
Figure 4. Formula for Recall at Rank r.....	10
Figure 5. Formula for Interpolated Precision.....	10
Figure 6. Formula for Average Interpolated Precision.....	11
Figure 7. Formula for Mean Average Interpolated Precision.....	11
Figure 8. Formula for Mean Average Interpolated Precision.....	12
Figure 9. Formula for Mean Average Interpolated Precision.....	13
Figure 10: Steps in Element Retrieval.....	16
Figure 11: Sample Document	18
Figure 12: Article Parse of Sample Document.....	18
Figure 13: Para + Mt Parse of Sample Document.....	19
Figure 14: Sample Article Parsing Configuration File.....	19
Figure 15: Lnu Element Vector Weighting Formula.....	23
Figure 16: Ltu Query Vector Weighting Formula.....	24
Figure 17: Sample Doc Tree.....	25
Figure 18: Sample Configuration File for Flex.....	26
Figure 19: Flex Output File.....	27
Figure 20: Sample XML file.....	29
Figure 21: FOL Output Sample File.....	30
Figure 22. Output File Produced After Evaluation.....	31
Figure 23. All-Element Retrieval Steps.....	33

1. Introduction

Information Retrieval is the field of employing various techniques for retrieving useful and highly relevant information from large collections of documents. In recent years, there has been an explosive growth in the amount of data being generated. Much of it can be attributed to the ever growing nature of the World Wide Web, and the information being stored can only be expected to increase. This makes increasingly complicated the task of locating the precise content that the user is seeking. Search engines like Google and Bing and many more are constantly working in the field of Information Retrieval to improve the accuracy of results when a user goes online to search for information.

XML is one of the most popular forms of textual data representation on the web. To provide the user with relevant and precise information in response to a query, one must consider the large amount of textual data represented as XML. The focus of this research is to extend traditional information retrieval, which concentrates on retrieving entire documents, to the retrieval of textual elements represented as XML, in the context of the INEX Ad-hoc task.

The research endeavors to improve XML retrieval systems by reducing the granularity of search results from entire documents to individual elements such as paragraphs, sections, etc., as well as enabling search within XML documents that do not strictly follow a DTD (Document Type Definition). We call these XML documents semi-structured documents.

This thesis describes the work performed within our research group at the University of Minnesota Duluth. The group is a participant in the INEX (Initiative for the Evaluation of XML Retrieval) competition [4]. This competition is sponsored by INEX to enhance the state-of-the art of XML retrieval. INEX provides all the participants an XML document collection (in 2009 a subset of Wikipedia), a set of queries, and metrics for evaluating the results returned. The performance of the participant systems is compared based on these metrics. As the Wikipedia document collection does not adhere strictly to a DTD, this research concentrates on retrieval from semi- structured documents.

The University Of Minnesota Duluth is a participant in the Ad Hoc retrieval track for INEX 2009. This thesis deals with this track in general and with the Focused Retrieval Task in particular. The tasks, along with the document collection, the evaluation measures, and the basics of the retrieval system are described in Chapter 2. Chapter 3 describes our Flexible retrieval system, (i.e., efficient element retrieval). Chapter 4 details our experiments in element retrieval and presents and analyzes the results of these experiments. Conclusions and suggestions for future work are discussed in Chapter 5.

2. Overview

In this chapter, we discuss the structure of INEX, including the retrieval tracks and the various evaluation measures used to measure the performance of the system. The Vector Space Model is also discussed in this chapter. It lies at the core of the Smart retrieval system [9], which is the basic retrieval system used in this research.

2.1 INEX

INEX stands for the **IN**itiative for **E**valuation of **X**ML. It is a competition which calls for development of various strategies for the purpose of retrieving XML documents and their contents. Various organizations from all over the world, including corporations and graduate schools, develop their own retrieval systems and compare them against those developed by other participants. The participants also contribute to the construction of test collections. INEX provides large test collections as well as evaluation measures to evaluate the results. Various tracks that are prevalent in INEX are the Book track, Entity Ranking, Efficiency, Ad Hoc, Link-the-Wiki, XML mining, etc.

The University of Minnesota Duluth IR group participates in the Ad Hoc track of INEX. The Ad Hoc track consists of four sub-tasks: Focused, Relevance in Context (RiC), Best in Context (BiC) and Thorough. In the 2009 Ad Hoc track, a set of static XML documents are searched using a set of queries. The collection used is the 2009 Wikipedia collection. Each system returns XML elements that are assessed for relevance. Using the specified evaluation measures, certain scores are generated. These scores

serve as a common measure on the basis of which performance of the participants is compared.

2.2 Document Collection

There have been a number of changes in the document collection used over the years in the INEX experiments. Before 2006, the collection used was an IEEE corpus which strictly conformed to a Document Type Definition (DTD). In 2007, INEX changed the collection to a Wikipedia XML document collection. These documents contain untagged text, i.e., text which does not belong to any of the defined elements. Such documents do not conform strictly to a DTD, and hence they are called semi-structured documents. The 2007 document collection was approximately 5.6 GB in size. This collection was used for our experiments of the IR research group in 2006, 2007 and 2008. In 2009, INEX provided a new Wikipedia collection which is approximately 50.7 GB in size. It contains 2,666,190 articles with over 30,000 unique tags in it. This collection is being used currently (i.e. for 2009 and 2010). A sample document is seen in Figure 1.

2.3 Topic or Query Collection

INEX asks each participating group to create a set of queries and submit them to INEX. After collecting queries from all participants, INEX releases a list of queries to the participants. The query is in form called as Content Only + Structure (CO + S). A CO + S query contains following fields, as seen in Table 1. A sample query is shown in Figure 2.

```

<article
  <name> text </name>
  <body>
    text
    <section>
      <title> text </title>
      Text
      <ss1> text </ss2>
      <p> text </p>
      ...
    </section>
    <p> text </p>
    ...
  </body>
</article>

```

Figure 1: Structure of a Sample Wikipedia Document

Field	Description
<title>	Contains CO query definition
<castitle>	Contains Content and Structure (CAS) query definition
<description>	Contains natural language definition of the information needed
<narrative>	Gives the definition of relevance and irrelevance

Table 1: Details of a topic [4]

2.4 Relevance Assessment

In the context of INEX, relevance is the extent to which any component of a document matches the information in the requested topic. A manual assessment of a subset of the document collection is done for each topic.

```
<inex_topic query_type="CO+S">
<title>band freddy mercury</title>
<castitle>//group[about(../singer, Freddy Mercury)]</castitle>
<phrasetitle>"Freddy Mercury"</phrasetitle>
<description>Find information about the bands in which Freddie
Mercury played.</description>
<narrative>
I am a huge fan of Freddy Mercury and all his work, and was wondering
in how many different bands, groups, or line-ups did he perform. I am
aware of some of them (such as Queen, obviously) but I am sure there
must be more.

I conduct this search for my own personal interest of completing my
record collection, I am trying to get hold of a complete discography
of Freddy Mercury that would shed light on his evolution as an artist
during his different career stages.

To be relevant an result should discuss a band, group, or line-up
that included Freddy Mercury. Not relevant is information about other
occurrences of Freddy Mercury in the media.
</narrative>
</inex_topic>
```

Figure 2. Sample Query [4]

Each participating group assesses three topics. The assessments are then pooled to form the set of all documents judged “relevant” to the query. This set forms the basis for evaluating the participants’ results. The relevance assessments are produced using a tool called GPXRai [4]. (The Best Entry point for a document can also be specified). Using this tool, text can be marked as relevant to a particular query. This text is converted to a File Offset and Length (FOL) format and then used for evaluation of

results. File Offset refers to the starting point in the document and length refers to the number of characters of relevant text from the offset. A qrels file which contains the information needed for evaluation is provided by INEX.

2.5 INEX Retrieval Tasks

Following is a brief description of the Ad Hoc tasks at INEX:

Thorough Task

The aim of the Thorough Task is to find all relevant elements or passages ranked in order of relevance. It will be therefore the case that, due to the nature of relevance in XML retrieval (i.e., if a child element is relevant, so will be its parent, although to a greater or lesser extent), an XML retrieval system that has estimated an element to be relevant may decide to return all its ancestor elements. This means that runs for this task may contain a large number of overlapping elements. [4]

Focused Task

The scenario underlying the Focused Task is to return to the user, for each topic, a ranked list of elements or passages. The Focused Task asks systems to find the most focused results that satisfy an information need, without returning "overlapping" elements. That is, for a given topic, no retrieval result in the result set may contain text already contained in another result. Or, in terms of the XML tree, no element in the result set should be a child or descendant of another element in the set. [4].

Relevant In Context Task

The scenario underlying the Relevant in Context Task is to return the relevant information (captured by a set of elements or passages) within the context of the full article. An article devoted to the topic of a request may contain a lot of relevant information across many elements.

The Relevant in Context Task asks systems to find a set of results that corresponds well to all the relevant focused elements in each article. For details, see [8].

Best in Context Task

The scenario underlying the Best in Context Task is to find the best entry point for starting to read an article (i.e., the point in the text where one should begin reading to find the relevant information sought). As a result, even an article completely devoted to the topic of request will only have one best entry point. The Best in Context Task asks systems to find the XML element or passage that corresponds to the best entry point. For details, see [12].

2.6 Evaluation Metrics

Thorough Task

Mean average interpolated precision (*MAiP*) score is used to evaluate Thorough task. The formula of *MAiP* is derived using the set of formulas given below.

Precision at rank r is defined as shown in Figure 3. Recall at rank r is defined as shown in Figure 4. The interpolated precision at recall value x is shown in Figure 5, average interpolated precision is calculated as shown in Figure 6, and the formula for calculating mean average interpolated precision is shown in Figure 7.

$$P[r] = \frac{\sum_{i=1}^r rsize(p_i)}{\sum_{i=1}^r size(p_i)}$$

where,

$P[r]$ is precision at rank r

p_r is the document part assigned to rank r in ranked list of documents

$rsize(p_r)$ is the length of relevant text contained by p_r in characters

$size(p_r)$ is the total number of characters contained by p_r

Figure 3. Formula for Precision at rank r

Focused Task

The result of the Focused task is evaluated using the metric *interpolated precision*, or more precisely interpolated precision at 1% recall, i.e., $iP[0.01]$. *Recall* is defined as the fraction of highlighted text that is retrieved and precision is defined as the fraction of retrieved text that is highlighted. The formula for interpolated precision ($iP[x]$) is given in the Thorough task above where x is replaced by a value of 0.01.

$$R[r] = \frac{\sum_{i=1}^r rsize(p_i)}{T_{rel}(q)}$$

where,

$R[r]$ is the recall score at rank r

p_r is the document part assigned to rank r in ranked list of documents

$rsize(p_r)$ is the length of relevant text contained by p_r in characters

$T_{rel}(q)$ is the total amount of relevant text for topic q

Figure 4. Formula for Recall at Rank r

$$iP[x] = \begin{cases} \max_{1 \leq r \leq |L_q|} (P[r] \wedge R[r] \geq x) & \text{if } x \leq R[|L_q|] \\ 0 & \text{if } x > R[|L_q|] \end{cases}$$

where,

$P[r]$ is the precision at rank r

$R[r]$ is the recall score at rank r

L_q is the ranked list of document parts returned by a retrieval system for a topic q

Figure 5. Formula for Interpolated Precision

$$AiP = \frac{1}{101} \cdot \sum_{x=0.00,0.01,\dots,1.00} iP[x]$$

where, $iP[x]$ is the interpolated precision at recall value of x

Figure 6. Formula for Average Interpolated Precision

$$MAiP = \frac{1}{n} \sum_t AiP(t)$$

where,

n is the number of topics,

$AiP(x)$ is the average interpolated precision for recall value of x

Figure 7. Formula for Mean Average Interpolated Precision

Relevant In Context Task

The results of this task are evaluated using generalized precision and recall, where the per document score reflects how well the retrieved text matches the relevant text in the document. Mean average generalized precision ($MAgP$) is used to get an overall performance estimate. [4].

Following formulas show how the $MAgP$ formula is derived. The formula

for generalized precision is shown in Figure 8 and the formula for calculating average generalized precision is shown in Figure 9. The *MAgP* score is simply the mean of the average generalized precision scores over all topics. [6]

$$gP[r] = \sum_{i=1}^r S(d_i)$$

where,

$gP[r]$ is the generalized precision,

r is the document rank,

$S(d_i)$ is the document score which varies between 0 (document without relevant text or none of the relevant text is retrieved) and 1 (all relevant text is retrieved without retrieving non-relevant text)

Figure 8. Formula for Generalized Precision

Best In Context Task

The results of this task are evaluated using generalized precision and recall. Mean average generalized precision (*MAgP*) is used to get an overall performance estimate. The formula for calculating *MAgP* is given above.

$$AgP = \frac{\sum_{r=1}^{|L|} IsRel(d_r) \cdot gP[r]}{N_{rel}}$$

where,

AgP is average generalized precision,

r is document-rank,

L is the ranked list,

$IsRel(d_r) = 1$ if document d at document rank r contains highlighted relevant text,

$IsRel(d_r) = 0$ if document d at document rank r does not contain highlighted relevant text,

$gP[r]$ is generalized precision, as calculated from formula above

Figure 9. Formula for Average Generalized Precision

2.7 Retrieval Model

Vector Space Model

The Vector Space Model [10] is the retrieval model used in this research. In this model, both the document and the query are represented as n -dimensional vectors, whose components are the terms (word types) occurring in them. These terms are weighted based on their frequency

within the document, and the distance between two vectors in vector space gives the correlation of a document to a query.

Smart Retrieval Engine

The retrieval engine we use is Smart [9]. It is based on the Vector Space Model. It has an automatic indexing component that constructs a vector representation of each document and the query. Retrieval produces a rank ordered list of documents (elements) that correlate with the query based on a specified measure. The vectors are weighted using Singhal's *Lnu-ltu* weighting scheme. This makes use of two collection-specific parameters called slope and pivot. [11]

The pivot is calculated as the average number of unique terms in a document (computed across the entire collection), and the value of slope is found experimentally. The weighting attempts to ensure that length disparities between element vectors do not bias results unfairly in favor of the longer elements.

Smart is used to index the collection and also for retrieval. It provides a utility called *smart-eval* which can be used to evaluate retrieval runs in terms of precision and recall. Smart produces the following files: *textloc file* (this file contains information about the physical location of the documents being indexed), *dictionary* (containing information about the terms in the documents and the document frequency of each term), *document vectors*, and an *inverted file* (for each term, a list of the document vectors that contain that term). Although Smart provides an

evaluation tool that evaluates the results, we use a tool provided by INEX. Though Smart acts as our retrieval engine for articles, we use Flex [7] for element retrieval.

3. Element Retrieval

This chapter gives a background for the element retrieval. It includes a discussion on the Flex retrieval system for element retrieval. Figure 10 displays the steps involved in element retrieval.

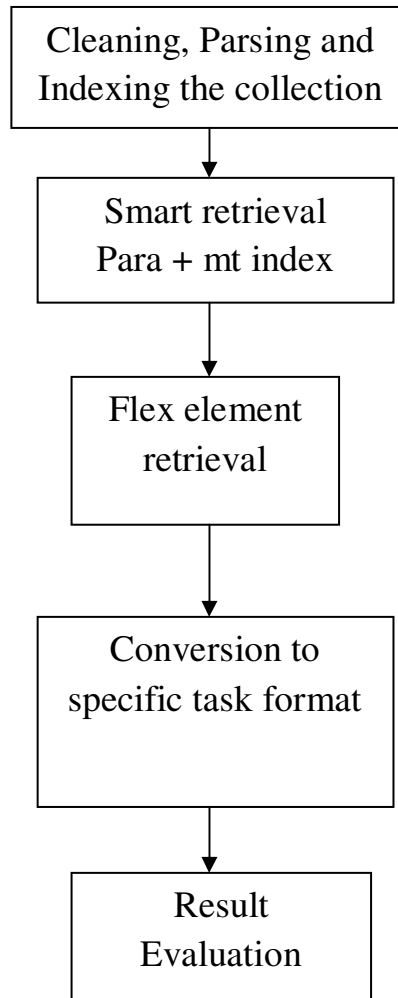


Figure 10. Steps in Element Retrieval

3.1 Cleaning, Parsing and Indexing

Cleaning

Cleaning the document collection ensures that for each starting tag in every document, there is a corresponding closing tag. All the unwanted tags are removed from the document, but the text in those tags is retained. Cleaning ensures that the collection is in good shape for the next stage of parsing.

Parsing

Parsing is the process of recognizing the entire text within a matching set of open and closed XML tags. The documents are parsed with respect to the articles (article parse) and paragraphs (para + mt parsing). In the case of article parsing, the entire set of text between the article tags is taken as one entity, thus producing only one element per article tag. In this process, all the contained tags inside article tag are removed.

In the paragraph parsing, all the defined terminal nodes of the XML document are identified. These terminal nodes are also called leaf nodes. In this parse, the text within these leaf nodes is identified (this includes the text contained inside leaf node tags). This parse also identifies the untagged text [5] in the document and marks it with an *mt* (magic text) tag at the level of its parent. The magic text is enclosed within the <mt> and </mt> tags. The text contained in the magic text tag cannot be discarded because the document building procedure done by Flex proceeds from the terminal nodes. And the text contained in all its child nodes must be

known to create the parent node. However, *mt* as a tag is never returned as a result.

Figure 11 shows a sample document containing text in the terminal element (i.e., `<p>`) and also untagged text.

```
<article>
<body>
  UMD
  <p> University of Minnesota Duluth, Kirby Drive 55812 </p>
  <p> Department of Computer Science </p>
</body>
</article>
```

Figure 11. Sample Document

Figure 12 shows the result of *article* parsing. As discussed above, a single element is produced containing all the text within the article tag.

```
UMD University of Minnesota Duluth, Kirby Drive 55812 Department of
Computer Science
```

Figure 12. Article Parse of Sample Document

The result of *para + mt* parsing is shown in Figure 13.

```
<p>
University of Minnesota Duluth, Kirby Drive 55812 Department of
Computer Science
</p>

<mt>
      UMD
</mt>
```

Figure 13. *Para + mt* Parse of Sample Document

A configuration file needs to be specified for the carrying out parsing. It supplies meta-information about the tags to be retained as well as tags to be indexed. Figure 14 shows a sample configuration file for article parsing.

```
tags_to_keep
article,body
tags_to_index
body
```

Figure 14. Sample Article Parsing Configuration File

The *tags_to_keep* represents all the tags listed as *tags to keep* and *tags_to_index* represent all the tags listed as *tags to index*. These are discussed in the section below.

Tag Sets

The XML documents contain tags which form the basis for selecting an element. We discuss here which tags are considered important. The tags are divided into two sets, namely tags to keep and tags to index.

1. *tags to keep* – These tags are considered important from the point of view of retrieval. Only elements listed as tags to keep are recognized as elements and returned as elements. Table 2 shows the tags to keep.

Tag Name	Tag representation
Article	<article>...</article>
Section	<section>...</section>
Sub-Section	<ss1>...</ss1>, <ss2>...</ss2>
Body	<body>...</body>
Paragraph	<p>...</p>
Emphasis	<emph3>...</emph3>
Normal-list	<normallist>...</normallist>
Ordered-list	...
Unordered-list	...
Number-list	<numberlist>...</numberlist>
Definition-list	<definitionlist>...</definitionlist>
Figure	<figure>...</figure>
Name	<name>...</name>
Title	<title>...</title>
Magic Text	<mt>...</mt>

Table 2. Tags to Keep

2. *tags to index* – All the terminal nodes are included in this set. (Thus none of these elements have child nodes). These tags are found in the *para + mt* parse. All the elements in this set are, loosely speaking, considered to be *paragraphs*. The set of *tags to index* are subset of *tags to keep*. Table 3 shows the tags to index – terminal node element tags.

Tag Name	Tag representation
Sub-Section	<ss1>...</ss1>, <ss2>...</ss2>
Paragraph	<p>...</p>
Emphasis	<emph3>...</emph3>
Normal-list	<normallist>...</normallist>
Ordered-list	...
Unordered-list	...
Number-list	<numberlist>...</numberlist>
Definition-list	<definitionlist>...</definitionlist>
Figure	<figure>...</figure>
Name	<name>...</name>
Magic Text	<mt>...</mt>
Table	<table>...</table>

Table 3. Tags to Index

Indexing

Indexing is done with respect to the various parses that are generated (e.g. article, section and *para + mt* parses). Taking the elements contained in the parse as input, indexing creates element and the query vectors.

Indexing is done by the Smart system. Smart generates term frequency

vectors called *nnn vectors*, with each vector representing an element. Indices are generated using the *article* parse and *para + mt* parse, respectively, as input. The article index is used for article retrieval whereas the *para + mt* index is used for terminal node retrieval which forms the basis for the Flex retrieval step. The configuration file used for the generating *para + mt* parse is the one used in this stage.

3.2. Smart Retrieval

The Smart engine is used for retrieval. A scheme known as *Lnu-ltu* weighting [11] is used in our research. In traditional weighting schemes, there is a tendency to produce bias towards longer elements with more terms and terms with higher frequency. Document vectors are *Lnu*-weighted and query vectors are *ltu*-weighted. *Lnu* is used to convert an *nnn*-weighted element vector to a corresponding *Lnu*-weighted vector.

The similarity score between an element and a query is found by taking the inner product of the *Lnu* weighted element vector with the *ltu* weighted query vector, and a ranked list of elements and/or articles is produced based on this score for each query. Figure 15 shows the *Lnu* term weighting formula. Formula for *ltu* query term weighting is given in Figure 16.

3.3 Flexible Retrieval (Flex)

Flex is the software component that implements flexible retrieval. It is written in C++. Flex generates the correlation scores for all elements with respect to the query. Flex also dynamically generates the document trees of interest.

$$\frac{1 + \log(tf)}{1 + \log(\text{avg } tf)}$$

$$(1 - \text{slope}) + \text{slope} * (\text{no. of unique terms})/\text{pivot}$$

where *tf* is the term frequency (as in nnn vector)

avg tf is the average term frequency of all terms in this vector

no. of unique terms is the number of distinct terms in this vector

slope is an empirically determined constant

pivot is the average length of the element vectors

Figure 15. Lnu Element Vector Weighting Formula

Seeding Flex:

This is the first step in flexible retrieval. For seeding Flex, retrieval against the *paragraph + mt* index is used. All of the elements of each document tree must be retrieved in order to generate the doc tree properly.

The seeding process requires following set of inputs:

1. Paragraph + mt retrieval.

$$\frac{1 + \log(tf) * \log(N/n_k)}{(1 - \text{slope}) + \text{slope} * (\text{no. of unique terms})/\text{pivot}}$$

where *tf* is the term frequency
N is the collection size
n_k is the number of documents that contain this term
slope and pivot are empirically determined constants
no. of unique terms is the number of distinct terms in this vector

Figure 16. Ltu Query Vector Weighting Formula

2. A numeral, *n*, which indicates the number of top ranked elements used to seed Flex.
3. Document schemas called doc trees. This specifies the structure of the documents that will be generated by Flex.
4. Docid-docpath mapping from the *paragraph + mt* retrieval. This mapping associates the XPath of each terminal node element identified in the schema of the seeded document with its corresponding Smart identifier.

A sample doc tree is shown in Figure 17.

```
/article[1]/body[1]/section[1] 0 0
/article[1]/body[1]/section[1]/emph3[1] 0 1
/article[1]/body[1]/section[1]/p[1] 0 0
/article[1]/body[1]/section[2] 3 1
/article[1]/body[1]/section[2]/title[1] 0 1
/article[1]/body[1]/section[2]/mt[1] 0 1
/article[1]/body[1]/section[3] 0 0
/article[1]/body[1]/section[3]/title[1] 0 1
/article[1]/body[1]/section[3]/normallist[1] 0 0
/article[1]/body[1]/section[4] 2 1
/article[1]/body[1]/section[4]/title[1] 0 1
/article[1]/body[1]/section[4]/normallist[1] 0 0
/article[1]/body[1]/section[5]/title[1] 0 1
```

Figure 17. Sample Doc Tree

The process of seeding, when completed, gives as output a set of doctrees with all of their terminal nodes populated (i.e., the content of each node is specified) for each query. Each terminal node in the doctrees is populated with its Smart id, which is nothing more than a pointer to its vector.

The rank-ordered list of *paragraph + mt* is used to locate all terminal nodes of the articles of interest. The docid-docpath mapping file is used to populate the contents of the terminal node elements. The calculation of the correlation scores is done by Flex for all the terminal node elements. The next step is to produce the seed subsets. Seed subsets are sets of seeded doctrees. In these doctrees, each terminal node is populated with its content representing the subset of documents retrieved by the query. These subsets are generated by filtering of doctrees based on the article retrieval. Each of these subsets contains complete document schemas with

all their terminal nodes populated. These subsets also form the input to flexible retrieval.

Flex Retrieval

Flex takes a configuration file as input as shown in Figure 18.

```
1. DOC_INDEX_PATH: path to the doc.nnn file of the paragraph + mt indexing
2. QUERY_INDEX_PATH: path to the query.nnn file of the paragraph + mt indexing
3. OUTPUT_PATH: path to the output produced by Flex
4. TREES_PATH: path to the output of seeding
5. NUM_OUTPUT_ELEMS: Number of elements in the ranked list per query
6. SLOPE_PARA: slope value used for paragraph + mt
7. SLOPE_SEC: slope value used for sections
8. SLOPE_BDY: slope value used for body
9. SLOPE_ALLELEMS: slope value of all elements
10. PIVOT_PARA: pivot value used for paragraph + mt
11. PIVOT_SEC: pivot value used for sections
12. PIVOT_BDY: pivot value used for body
13. PIVOT_ALLELEMS: pivot value of all elements
14. N_PARA_VALUES: Number of paragraphs. Can get this value from the textloc.txt file of the paragraph + mt indexing
15. N_SEC_VALUES: Number of sections. Can get this value from the textloc.txt file of the section indexing
16. N_BDY_VALUES: Number of articles. Can get this value from the textloc.txt file of the article indexing
17. N_STATS_FOLDER: path to the paragraph + mt indexing
```

Figure 18. Sample Configuration File for Flex

Flexible retrieval is performed after seeding. The seed subsets produced are taken as input by Flex. A parent vector is generated using the content of all the terminal nodes and the correlation score between the parent vector and query is calculated. A bottom-up approach is taken in which the score calculation starts at the leaf nodes of the doc tree and moves to the root of the tree. This bottom up approach is taken because Flex cannot calculate the score of a parent if a child's content is not known. The result is a ranked list of elements for each document.

The output of Flex contains each element along with its correlation score. Final output contains, for each query, the elements in a rank-ordered list. A sample Flex output file is shown in Figure 19.

```
5 645414/article[1]/body[1]/section[1] 19.8346
5 60743/article[1]/body[1]/section[3]/ss1[1] 19.4672
5 28342/article[1]/body[1] 19.1136
5 60743/article[1]/body[1]/section[3]/section[7] 18.8655
5 60743/article[1]/body[1]/section[6] 27.2642
5 682628/article[1]/body[1] 18.344
5 60743/article[1]/body[1]/p[1] 18.0234
```

Figure 19. Flex Output Sample File

Post-processing

Once the rank-ordered list of elements is produced, some post-processing needs to be done. The magic text tags which were introduced to account for the untagged text must be removed since they are not legal tags. Hence all the XPath entries containing mt tags are removed from the rank ordered list that Flex produces.

3.4 Conversion to Specific Task Format

As output of the Flexible retrieval, a rank-ordered list is generated that consists of m elements from the n retrieved documents. Also present is the correlation score of each of the elements. The rank-ordered list must be filtered to meet the pre-requisites of the Focused task. The focused task expects a ranked list of elements that is non-overlapping. Overlap can be removed using one of the three strategies: *child*, *section* and *correlation*. See [1] for details

3.5 Result Evaluation

The focused result that is obtained with the help of one of the strategies can now be evaluated. Evaluation is done with the help of a tool provided by INEX. There are three steps in the evaluation process:

1. The result files are converted to XML format. This is done with the help of a script provided by INEX [4].
2. In the second step, the XPath of the returned elements are converted into File Offset and Length (FOL) format using a package provided by INEX. A sample FOL file is shown in Figure 14.
3. The evaluation tool is then used to produce the evaluations. This tool is a software package written in Java. The tool calculates the scores using the corresponding metric for each task. The input to this tool are the qrels file provided by INEX, the FOL formatted file produced in the previous step, and a file that contains File Offset Length of every element in the collection (again provided by INEX). The qrels file is

generated from manual relevance assessments and is used as the basis for evaluating results.

A sample XML file is shown in Figure 20, a sample FOL file is shown in Figure 21, and Figure 22 shows the output file after evaluation.

```
<inex-submission participant-id="72" run-id="focused"
task="Focused" query="automatic" result-type="element"> <topic-
fields title="yes" mmtitle="no" castitle="no" description="no"
narrative="no"/> <description></description>
                                <collections>
<collection>wikipedia</collection>
</collections>
<topic topic-id="1">
  <result>
    <file>985414</file>
    <path>/article[1]/body[1]/section[1]</path>
    <rank>1</rank>
  </result>
  <result>
    <file>20347</file>
    <path>/article[1]/body[1]/section[6]</path>
    <rank>2</rank>
  </result>
</topic>
</inex-submission>
```

Figure 20. Sample XML File

```
3 Q0 985414 1 -1 umd-focused 393 785
3 Q0 20347 2 -1 umd -focused 13174 2365
3 Q0 20347 3 -1 umd -focused 15539 1658
3 Q0 20347 4 -1 umd -focused 23262 1670
3 Q0 20347 5 -1 umd -focused 74 637
3 Q0 20347 6 -1 umd -focused 2344 3475
3 Q0 233013 7 -1 umd -focused 721 2076
3 Q0 1654632 8 -1 umd -focused 622 1400
3 Q0 20347 9 -1 umd -focused 21806 1456
3 Q0 261763 10 -1 umd -focused 1405 727
```

Figure 21. Sample FOL File

```
<eval run-id="umd_focused" file="/smart/  
/flex/focused_section/xml/25/focused_50_100.xml">  
  
num_q all 104  
  
num_ret all 6664145  
  
num_rel all 11614635  
  
num_rel_ret all 1310084  
  
iP[0.00] all 0.4950613331  
  
iP[0.01] all 0.4501345937047  
  
iP[0.05] all 0.336459493028904  
  
iP[0.10] all 0.259673978169276  
  
MAiP all 0.10435127000252057  
  
ircl_prn.0.00 all 0.51647524061333  
  
ircl_prn.0.01 all 0.470123208937047
```

Figure 22. Output File Produced After Evaluation

4. Experiments, Results, and Analysis

This chapter provides the details of the experiments performed. The goal of these experiments is to determine the most effective strategies for producing top-ranked results for the task in question. It also includes a brief background of each experiment and the results produced. All these experiments were performed on the INEX 2009 collection, a set of XML documents with a total size of approximately 50 GB.

4.1 All-element retrieval

All-element retrieval is considered a base case. The set of 1500 top ranked elements from all-element retrieval (using the “best value” of slope and pivot) is used as a baseline for these experiments. An all-element index is used to generate this base case. The flowchart for carrying out the all-element retrieval is given in Figure 23.

4.2 Thorough Task

Definition: To retrieve all elements along the path of a retrieved element. For carrying out the Thorough task, article retrieval is performed to identify the set of top-ranked articles. Flex is seeded only with the highly correlating elements from the articles in this set. The steps involved in carrying out the Thorough task experiments are as follows:

1. Article Retrieval – A ranked list of n articles is retrieved.

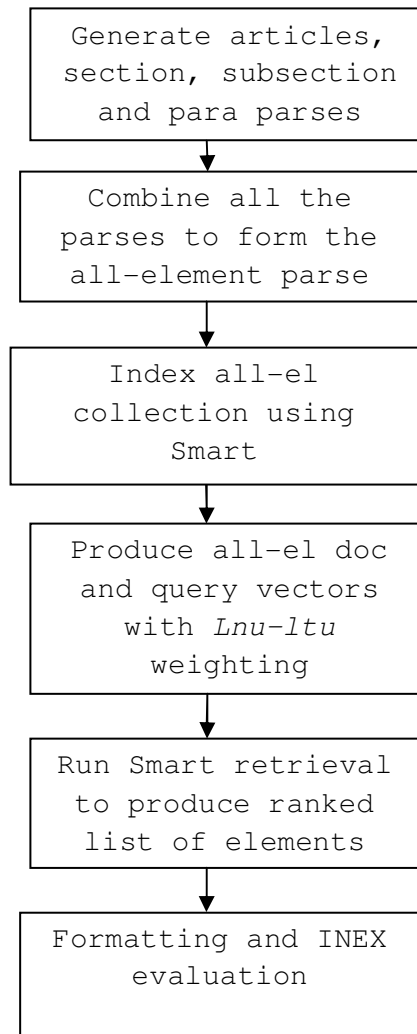


Figure 23. All-element Retrieval Steps

2. Flex builds the doc trees for these articles. *Lnu-ltu* weighting is used by Flex; it uses the specified value of slope (0.11).
3. Flex produces, for each document, a rank ordered list of (overlapping) elements.
4. These elements are reported in article order.

5. After filtering the elements retrieved in step 4 and removing all the nodes tagged by mt, we get the output for Thorough task.

In this task, a ranked-ordered list of all positively-correlating elements is expected. Thorough task results are seen in Tables 4, 5, and 6.

Tables 4, 5 and 6 show the 2009 Thorough task MAiP scores for the slope values of 0.10, 0.11 and 0.12 respectively.

Number of elements → ----- Number of articles ↓	50	100	150	200	250	500	1000	1500
25	0.2014	0.2026	0.2029	0.2029	0.2029	0.2029	0.2029	0.2029
50	0.2015	0.2027	0.2031	0.2036	0.2036	0.2036	0.2036	0.2036
100	0.2016	0.2028	0.2032	0.2038	0.2044	0.2044	0.2044	0.2044
150	0.2017	0.2029	0.2031	0.2039	0.2045	0.2048	0.2048	0.2048
200	0.2018	0.2031	0.2033	0.2041	0.2046	0.2049	0.2049	0.2049
250	0.2018	0.2032	0.2035	0.2041	0.2047	0.2051	0.2052	0.2052
500	0.2019	0.2032	0.2036	0.2042	0.2048	0.2051	0.2053	0.2054

Table 4. Thorough Task Evaluation for Slope=0.10, Pivot = 44 (2009)

We find that the best results are obtained when we use a slope value of 0.11. This value of slope is used to execute the significance tests [2] which indicate how the results of UMD compare with those of the rest of participants at INEX.

Number of elements → ----- Number of articles ↓	50	100	150	200	250	500	1000	1500
25	0.2062	0.2069	0.2081	0.2081	0.2081	0.2081	0.2081	0.2081
50	0.2062	0.2071	0.2081	0.2085	0.2085	0.2085	0.2085	0.2085
100	0.2063	0.2071	0.2082	0.2086	0.2095	0.2095	0.2095	0.2095
150	0.2063	0.2073	0.2084	0.2087	0.2096	0.2098	0.2098	0.2098
200	0.2064	0.2074	0.2083	0.2088	0.2098	0.2099	0.2105	0.2105
250	0.2064	0.2075	0.2085	0.2091	0.2097	0.2101	0.2109	0.2109
500	0.2065	0.2077	0.2085	0.2091	0.2098	0.2103	0.2109	0.2120

Table 5. Thorough Task Evaluation for Slope=0.11, Pivot = 44 (2009)

Number of elements → ----- Number of articles ↓	50	100	150	200	250	500	1000	1500
25	0.2011	0.2016	0.2028	0.2028	0.2028	0.2028	0.2028	0.2028
50	0.2012	0.2018	0.2029	0.2035	0.2035	0.2035	0.2035	0.2035
100	0.2013	0.2021	0.2031	0.2036	0.2044	0.2044	0.2044	0.2044
150	0.2014	0.2022	0.2030	0.2035	0.2043	0.2046	0.2046	0.2046
200	0.2013	0.2023	0.2029	0.2036	0.2043	0.2049	0.2051	0.2051
250	0.2015	0.2024	0.2031	0.2037	0.2045	0.2051	0.2053	0.2053
500	0.2016	0.2024	0.2032	0.2038	0.2045	0.2051	0.2058	0.2061

Table 6. Thorough Task Evaluation for Slope=0.12, Pivot = 44 (2009)

Observations

The observations from the experiments are indicated below:

1. For the Thorough Task, the best MAiP score (**0.2120**) is produced by a slope value of 0.11. It is observed that Focused task results are superior for this slope value. [13].

2. With the results obtained in these experiments, the rank of University of Minnesota Duluth increases to 8th overall from 12th in the 2009 Thorough Task of Ad-hoc track, indicating a jump of 4 places. Our earlier (baseline) Thorough task score was low primarily due to problems inherent in parsing. The changes made in parsing for the 2009 collection (see [13]) made the improvements possible.

Table 7 shows the improvement in the results compared to last year.

Participant ID	MAiP	Institute
P48	0.2855	LIG
P6	0.2818	Univ. of Amsterdam
P5	0.2585	Queensland Univ. of Tech.
P92	0.2496	Univ. of Lyon
P60	0.2435	Saint Etienne University
P346	0.2350	Univ. of Twente
P10	0.2133	Max-Planck-Institute
P72 (Thorough)*	0.2120	UMD
P72 (All Element)*	0.1920	UMD
P167	0.1390	School of Ele. Engg. & CS
P68	0.0630	Univ. of Pierre
P25	0.0577	Renmin Univ. of China
P72 (Thorough)**	0.0562	UMD

* Current UMD results compared to official runs.

** UMD results in 2009

Table 7. Top-ranked Participants in Thorough Task – 2009

5. Conclusions and Future Work

Conclusions

We have established a slope value which when used in Thorough Task gives a good result in terms of rank order and significance. For the Ad-hoc track, our results are as good as the best results; i.e., we have good results for the Thorough, Focused and RIC tasks.

Future Work

We know a lot about Ad-hoc track compared to previous years. However, it is difficult to tell which direction the Ad-hoc track is going to proceed starting next year. The IR research group at UMD might turn its attention to the Book Track next year; it presents interesting challenges.

References

- [1] Bhirud, D. Focused Retrieval using Upper Bound Methodology, MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2009. <http://www.d.umn.edu/cs/thesis/Bhirud.pdf>

- [2] Cherukuri R. Significance Testing for INEX 2008/2009 Ad-hoc track. MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2010. <http://www.d.umn.edu/cs/thesis/Cherukuri.pdf>

- [3] Geva S., Kamps J., Trotman A. INEX 2009 Workshop Pre-Proceedings. <http://www.inex.otago.ac.nz/data/proceedings/INEX2009/preproceedings.pdf>

- [4] INEX 2009 Ad-hoc track <http://www.inex.otago.ac.nz/tracks/adhoc/gtd.asp>

- [5] Kamat, N. Impact of Untagged Text in Dynamic Element Retrieval, MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2007. http://www.d.umn.edu/cs/thesis/Kamat_ms.pdf

- [6] Kamps, J., Pehcevski, J., Kazai, G., Lalmas, M., Robertson, S. INEX 2007 Evaluation Measures. *Focused Access to XML Documents*, Springer, LCNS 4862, 70-79, 2008.

- [7] Khanna, S. Design and Implementation of a Flexible Retrieval System. MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2005. <http://www.d.umn.edu/cs/thesis/Khanna.pdf>

- [8] Polumetla. C. Improving the Results of the Relevant in Context Task.

MS Thesis, Department of Computer Science, University of Minnesota Duluth, 2009. <http://www.d.umn.edu/cs/thesis/Polumetla.pdf>

- [9] Salton, G., ed. *The Smart Retrieval System – Experiments in Automatic Document Processing*. Prentice-Hall, 1971.
- [10] Salton, G., Wong, A., Yang, C. A Vector Space Model for Automatic Indexing, *Comm. ACM*, 18(11), 613-620, 1975.
- [11] Singhal, A., Buckley, C., Mitra, M. Pivoted Document Length Normalization, *Proc. Of the 19th Annual International ACM SIGIR Conference*, 19-21, 1996.
- [12] Sudhakar, V. Improving Results for the Best in Context Task. MS Project, Department of Computer Science, University of Minnesota Duluth, 2009.
- [13] Vadlamudi S. Producing improved results for the INEX Focused and Relevant in Context tasks. MS Thesis Department of Computer Science, University of Minnesota Duluth, 2010. <http://www.d.umn.edu/cs/thesis/Vadlamudi.pdf>