

UNIVERSITY OF MINNESOTA

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

Darshan D. Paranjape

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.

---

Name of Faculty Adviser

---

Signature of Faculty Adviser

---

Date

GRADUATE SCHOOL

# **Improving Focused Retrieval**

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

Darshan D. Paranjape

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

July, 2008

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



## **Acknowledgements**

I would like to thank Dr. Donald Crouch for giving me a wonderful opportunity to work in the field of information retrieval. He has immense knowledge in this field and I am really grateful to him for his continuous guidance and valuable feedback on my work.

I would also like to thank Dr. Carolyn Crouch for her valuable feedback. I would like to thank Aditya Mone and Nachiket Kamat for sharing their knowledge related to the flexible retrieval system. Special thanks to my colleague Salil Bapat for being supportive co-worker as well as good friend throughout the completion of this thesis.

I would like to thank the staff at the Department of Computer Science – Lori Lucia and Linda Meek for their continuous help. Special thanks to the system administrator – Jim Luttinen for helping me with the system related issues.

Finally I would like to thank my fiancée Monali Patil, my family members and all my friends for encouraging and supporting me throughout my life.

## **Abstract**

Traditional information retrieval strategies focus on retrieving information from unstructured text documents. But with the tremendous growth of World Wide Web and the advent of Extensible Markup Language (XML) as a preferred standard for storing web documents, the focus is now shifting to retrieval from structured and semi-structured documents. The XML markup partitions the document into well-defined and non-overlapping elements, thus giving it a tree-like structure. Our method of searching for information at various levels of this tree and returning highly correlating elements to the user is called *flexible retrieval (Flex)* [5].

A major objective of this thesis is to improve the performance of the INEX 2007 [3] Ad hoc Focused Task. The Focused Task requires retrieval systems to find the most focused elements that satisfy the information need expressed in the query. In this thesis, we discuss the use of flexible retrieval in focused retrieval. Overlap removal strategies which convert flexible retrieval results to focused results are explained. Experimental results show marked improvement through the use of a new set of terminal node tags and the incorporation of article retrieval with element retrieval.

# Table of Contents

List of Figures.....	v
List of Tables.....	vi
<b>Chapter 1. Introduction.....</b>	<b>1</b>
<b>Chapter 2. Background.....</b>	<b>3</b>
2.1 INEX (Initiative for Evaluation of XML retrieval).....	3
2.2 Document Collection.....	3
2.3 Query Collection.....	4
2.4 Relevance Assessments.....	5
2.5 INEX 2007 Ad hoc task.....	6
2.6 Evaluation Metric.....	7
2.7 Smart Retrieval Engine.....	8
<b>Chapter 3. The Flexible Retrieval System.....</b>	<b>9</b>
3.1 Pre-Flex Operations.....	10
3.2 Flex.....	14
3.3 Post-Flex Operations .....	16
3.4 Importance of Magic Text.....	18
<b>Chapter 4. Improving Focused Retrieval .....</b>	<b>20</b>
4.1 Focused Retrieval.....	20
4.2 Overlap Removal Strategies.....	21
4.3 2007 Results.....	24
4.4 Analysis.....	27

<b>Chapter 5. Incorporating Article Retrieval with Flexible Retrieval.....</b>	<b>29</b>
5.1 Importance of Article Retrieval.....	29
5.2 Method.....	29
5.3 Results.....	30
5.4 Analysis.....	32
<b>Chapter 6. Future Work.....</b>	<b>33</b>
<b>References.....</b>	<b>34</b>

## List of Figures

1. Structure of a typical Wikipedia document.....	4
2. Sample query.....	5
3(a). Procedure for pre-Flex operations.....	9
3(b). Procedure for Flex operations.....	10
3(c). Procedure for post-Flex operations.....	10
4. Lnu weighting formula.....	13
5. Ltu weighting formula.....	14
6. Typical semi-structured Wikipedia document.....	18
7. Completely structured document.....	19
8(a). Excerpt from Flex result file showing XPathS from a single article.....	22
8(b). Tree structure represented by XPpaths.....	22

## List of Tables

1(a). Tags identified as terminal nodes for 2007 (Set 1- Old Tags).....	11
1(b). Tags identified as terminal nodes for 2007 (Set 2- New Tags).....	11
2(a). Slope and Pivot values (Set 1- Old tags).....	13
2(b). Slope and Pivot values (Set 2- New tags).....	13
3. 2007 Focused results for old tags using correlation score strategy.....	25
4. 2007 Focused results for old tags using terminal node strategy.....	25
5. 2007 Focused results for new tags using correlation score strategy.....	26
6. 2007 Focused results for new tags using terminal node strategy.....	27
7. 2007 Focused results for old tags based on article retrieval.....	31
8. 2007 Focused results for new tags based on article retrieval.....	31

# 1. Introduction

Information retrieval is the science of searching for relevant information in a large collection of document-based data. This information is returned in response to a user's query. Information retrieval is a rapidly growing field, and web-based retrieval systems like Google, Yahoo Search and Live Search have become important components of daily web activities.

Traditional information retrieval strategies focus on retrieving information from unstructured text documents. Such systems return sets of "relevant" documents in response to the user's query. The user must browse through each document to find relevant portions of text. But with the tremendous growth of World Wide Web and the advent of Extensible Markup Language (XML) as a preferred standard for storing web documents, the focus is now shifting to retrieval from structured documents.

The use of XML enables the retrieval of *elements* from a document (rather than just the document as a whole). The XML markup partitions the document into well-defined and non-overlapping elements, thus giving it a tree-like structure. Our method of searching for relevant information at various levels of this tree and returning highly correlating elements to the user is called *flexible retrieval* [5].

The University of Minnesota Duluth (UMD) is a participant in the INEX (Initiative for the Evaluation of XML Retrieval) competitions. The main aim of INEX is to provide an infrastructure, in the form of large XML test collections and appropriate scoring methods, for the evaluation of XML-based information retrieval systems [3]. Participants can then compare the effectiveness of their XML retrieval system to that of others.

A major objective of this thesis is to improve the results of Focused retrieval. The Focused task is described in Chapter 2 and the experiments designed to achieve this goal are described in Chapter 4 and Chapter 5.

Chapter 1 presents introductory information. Chapter 2 presents an overview of the INEX 2007 tasks, document collection, queries (topics), relevance assessments and the Smart retrieval system. Chapter 3 describes flexible retrieval system developed at UMD. Chapter 4 presents the overlap removal strategies and use of new set of terminal node tags. The experiments designed to improve Focused Task results by incorporating article retrieval with element retrieval are presented in Chapter 5. Suggestions for future research are given in Chapter 6.

## **2. Background**

This chapter provides an overview of INEX, the 2007 INEX document collection and topics (or queries), the INEX 2007 Ad hoc task, relevance assessments and evaluation measures. It also describes the Smart experimental retrieval system.

### **2.1 INEX (Initiative for Evaluation of XML retrieval)**

INEX [3] is an initiative that facilitates the development of effective XML-based information retrieval strategies. Organizations participating in INEX competitions compare the effectiveness of their XML retrieval systems against those of other participants. By contributing to the construction of the XML test collections, they provide a means for future comparative experiments.

UMD participates in the Ad hoc track of INEX. The tasks involve searching a set of static XML documents using a defined set of topics or queries. The queries may contain both content and structural conditions. In response to a query, the system retrieves XML elements from the collection. The main purpose of the INEX 2007 Ad hoc track is to investigate the value of the internal document structure (as provided by the XML markup) for retrieving relevant information. By comparing the values produced when INEX evaluation measures are applied to the elements (or passages) retrieved by competing participants, the systems themselves are compared. [2]

### **2.2 Document Collection**

The INEX 2007 document collection is a set of Wikipedia documents in XML format. The Wikipedia XML Corpus is based on an early 2006 version of the English Wikipedia collection converted to XML format. The collection contains 659,338 Wikipedia articles. On average, an article contains 161 XML nodes, where the average depth of a node in the XML tree of the document is 6.72. [2]

The original Wiki syntax has been converted into XML, using both general tags of the layout structure (article, section, paragraph, title, list and item), typographical tags (bold, emphatic), and frequently occurring link-tags (collectionlink, unknownlink). The structure of a typical Wikipedia document is presented in Figure 1.

```
<article>
<name> text </name>
<body>
  text
  <section>
    <title> text </title>
    text
    <p> text </p>
    ...
  </section>
  <p> text </p>
  ...
</body>
</article>
```

**Figure 1: Structure of a typical Wikipedia document (INEX 2007)**

### **2.3 Query Collection**

The INEX 2007 ad hoc queries are created by participants following the formal guidelines for topic development. They are referred to as CO + S (Content Only + Structure) queries. They specify both content (information to be retrieved, found in the title field) and structural constraints (found in the castitle field). The INEX 2007 Ad hoc track contains 107 CO + S queries. Figure 7 presents a sample query.

```
<inex_topic topic_id="414" ct_no="3">

<title>hip hop beat</title>

<castitle>//*[about(., hip hop beat)]</castitle>

<description>what is a hip hop beat?</description>

<narrative>
To solve an argument with a friend about hip hop music and beats, I want to learn all
there is to know about hip hop beats. I want to know what is meant by hip hop beats,
what is considered a hip hop beat, what distinguishes a hip hop beat from other beats,
when it was introduced and by whom. I consider elements relevant if they specifically
mention beats or rhythm. Any element mentioning hip hop music or style but doesn't
discuss anything about beats or rhythm is considered not relevant. Also, elements
discussing beats and rhythm, but not hip hop music in particular, are considered not
relevant.
</narrative>

</inex_topic>
```

**Figure 2: Sample query (topic) [Topic id = 414]**

## **2.4 Relevance Assessments**

INEX provides relevance assessments against which results provided by participating organizations are evaluated. All participants contribute towards the creation of relevance assessments. Relevance assessments are produced by manual assessment of document elements against the query. For every query, INEX retrieves a number of articles using their own TopX retrieval engine and forms a pool of these articles. These article pools are provided to the manual assessor along with the query set. The relevance assessor must first understand the information requested by the query, then read the document and

highlight the text which he/she finds relevant to the topic. The relevance assessor is also asked to mark the best entry point for each article he/she has judged relevant.

## **2.5 INEX 2007 Ad hoc Tasks**

The INEX 2007 Ad Hoc Track features three tasks: Focused, Best-in-Context and Relevance-in-Context. (The Thorough task defined in INEX 2006 was eliminated in 2007.)

### **2.5.1 Focused Task**

The Focused Task requires retrieval systems to find the most focused results that satisfy the information need specified in the query. The main aim of this task is to return a ranked list of elements, where no element overlaps with any other element. The assumption of the Focused task is that smaller elements are more “focused” towards the query. Thus, if there is more than one element along a path with the same correlation score, the lower level (i.e., more precise) element is returned. In our implementation of the Focused task, we take the ranked list of elements produced by flexible retrieval and remove all overlapping elements, keeping the elements that best correlate with the query. We have two different strategies for removing overlap. In the first strategy, the element with the higher correlation score is given priority (i.e., is returned) if there is more than one element along the same path. In the second strategy, we return only the terminal node along a path (all other elements are removed).

### **2.5.2 Best-in-Context Task**

The purpose of this task is first to identify the articles relevant to the query and then to identify the best entry point (BEP). The BEP is that element which identifies the point where the user should begin reading to satisfy his/her information need. In our implementation of the Best-in-Context task, we take the ranked list of elements as the output of the flexible retrieval and return a single element from each article. We have several different strategies for selecting the single element from an article. In one strategy, the element with the highest correlation score is selected. Another approach

takes the ranked list of elements produced by flexible retrieval and selects an element from this list based on its physical position.[6]

### **2.5.3 Relevance-in-Context Task**

A relevant article is likely to contain relevant information that is spread across different elements. This task requires the retrieval system to find the relevant elements in each relevant article. In our implementation of the Relevance-in-Context task, we first identify the articles relevant to the query and then, for each of these articles, return highly correlating, non-overlapping elements (i.e., all “relevant elements associated with that article”).[1]

## **2.6 Evaluation Metric**

Evaluation measures used in INEX 2007 are completely different from those used in previous INEX competitions. Previously, results were element-based, since the element was the most natural unit of retrieval and the metric used was element-based. The main change in INEX 2007 is the inclusion of arbitrary passages of text. This change allows the retrieval format to include consecutive elements (or passages) as well as single elements of text. Hence the evaluation measures are now based directly on the highlighted text, as identified by the assessors. As a result it is now possible to compare systems retrieving elements or passages. [7]

The INEX 2007 measures are solely based on the retrieval of highlighted text (i.e., it is assumed that the retrieval systems return the highlighted text). For Focused and Relevance-in-Context tasks, the characters of text retrieved by the system are compared to the characters of text identified as relevant by the assessor. The Best-in-Context metric uses the distance between the best entry point returned by the system and that identified by an assessor as the basis for evaluation. [7]

In the Focused task, recall is measured as the fraction of all highlighted text that has been retrieved. Precision is measured as the fraction of retrieved text that was highlighted. The evaluation measure for the Focused task is interpolated precision at 1% recall (iP [0.01]).

The evaluation of the Relevant-in-Context task is based on the measures of generalized precision and recall, where the document score reflects how well the retrieved text matches the relevant text in the document. The evaluation measure for the Relevance-in-Context task is mean average generalized precision (MAgP).

The evaluation of the Best-in-Context task is based on the measures of generalized precision and recall, where the document score reflects how well the retrieved entry point matches the best entry point in the document. The evaluation measure for the Best-in-Context task is mean average generalized precision (MAgP).

## **2.7 Smart Retrieval Engine**

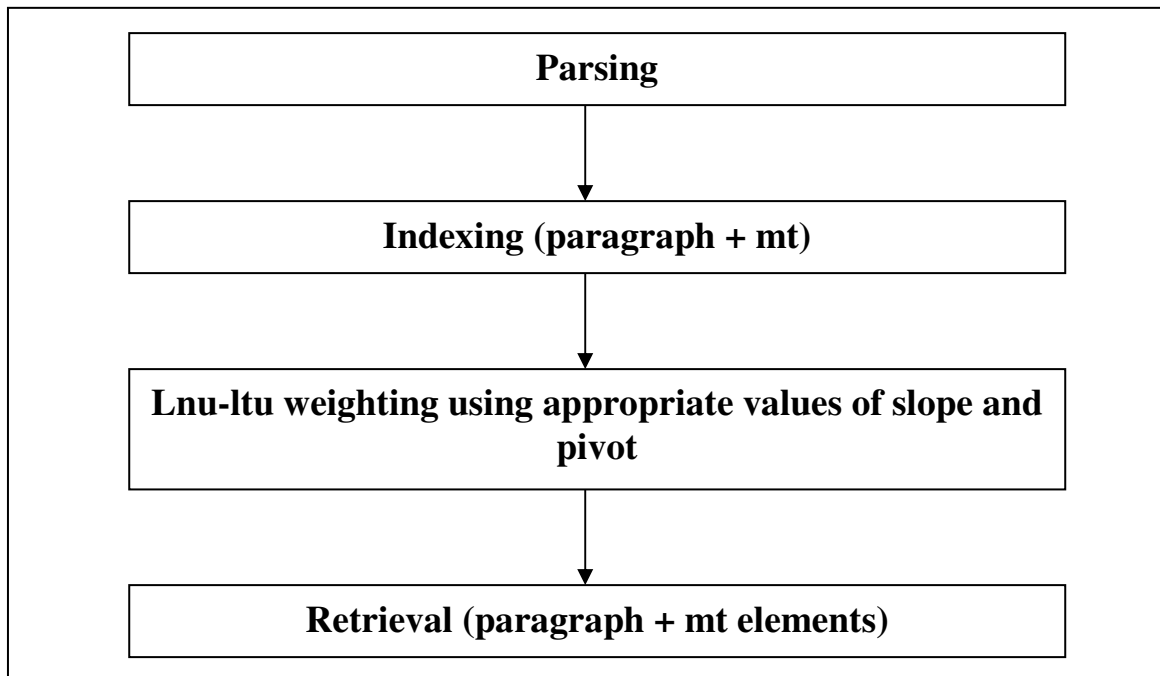
We use Smart 13.0 [8] as our basic retrieval engine. Smart uses the Vector Space Model [9], in which each document and query is represented as a weighted vector of terms. The distance between the document vector and the query vector in the vector space determines the correlation of the document with the query. Smart provides us with the basic functionalities of indexing the document collection, weighting the document and query vectors, and retrieving (highly correlating) document elements. [4]

### 3. The Flexible Retrieval System

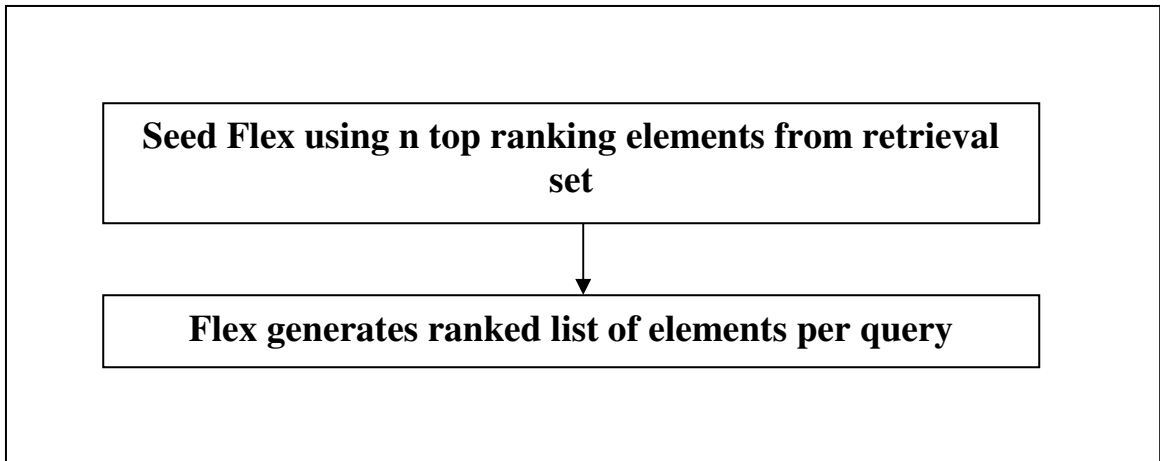
In this chapter, the procedure for flexible retrieval is presented. A detailed explanation of the stages involved in flexible retrieval (parsing, indexing, *Lnu-ltu* weighting, Flex operation and output processing) is presented. This chapter also describes the importance of magic text (mt) tags.

Flexible Retrieval refers to the task of retrieving specific elements in response to a query. Whereas traditional IR systems retrieve entire documents, our system of flexible retrieval seeks to recognize those elements that are specific to (i.e., focused on) the query, thereby reducing the time the user has to expend to find what he is looking for.[5]

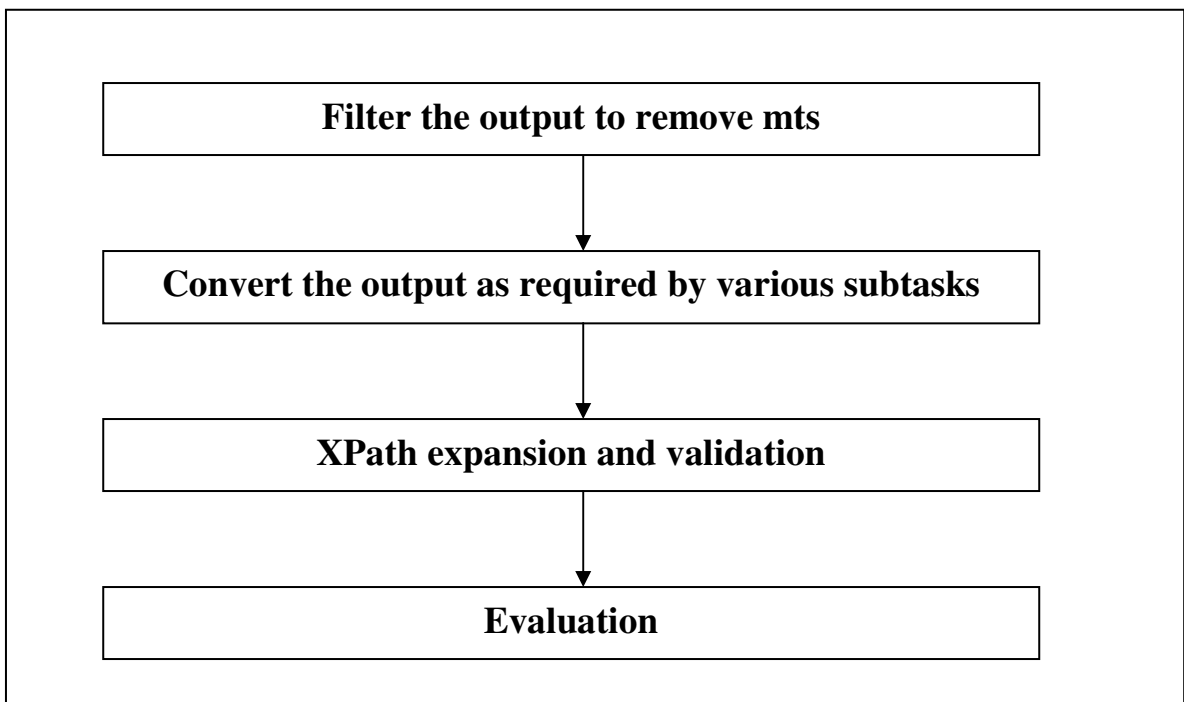
The procedure for flexible retrieval is given in Figure 3. Details of each step are described in the following sections.



**Figure 3(a): Procedure for pre-Flex operations**



**Figure 3(b): Procedure for Flex operations**



**Figure 3(c): Procedure for post-Flex operations**

### **3.1 Pre-Flex Operations**

Pre-Flex operations begin by parsing the document collection and indexing the terminal node elements. After indexing, we perform *Lnu-ltu* term weighting and retrieval against the terminal node index. A set of top-ranked elements (used as an input to Flex) are retrieved for every query.

## Parsing

The first step of flexible retrieval is to parse the document into constituent elements. We generate a paragraph-and-mt parse to create the paragraph-and-mt index required for flexible retrieval. (This parse generates all terminal nodes along with the magic text elements.) Once a terminal node tag is recognized in the XML hierarchy, any tags contained in it are removed and all the text is aggregated to form the leaf node. Table 1 presents two different sets of terminal node tags which were used in these experiments.

<b>Identified Terminal Elements</b>	<b>Terminal Tags</b>
Paragraph	<p>...</p>
Normal-list	< normallist >...</normallist>
Ordered-list	<ol>...</ol>
Unordered-list	<ul>...</ul>
Number-list	<numberlist>...</ numberlist >
Definition-list	<definitionlist>...</ definitionlist >
Figure	<figure>...</figure>
Table	<table>...</table>
Magic Text	<mt>...</mt>

**Table 1(a): Tags identified as terminal nodes for 2007 (Set 1- Old Tags)**

<b>Identified Terminal Elements</b>	<b>Terminal Tags</b>
Paragraph	<p>...</p>
Figure	<figure>...</figure>
Name	<name>...</name>
Emph	<emph3>...</emph3>
Magic Text	<mt>...</mt>

**Table 1(b): Tags identified as terminal nodes for 2008 (Set 2- New Tags)**

## **Indexing**

Parsing is followed by the creation of indices using Smart. We generate a paragraph-and-mt index by taking the paragraph-and-mt parse as input. Queries are also indexed along with the elements. The title part of every query is used as the text for query indexing.

## **Term Weighting and Pivoted Normalization**

Our Smart index of the leaf nodes uses the *nnn* weighting scheme [5] to weight terms in the element vector. In this weighting scheme, the weight assigned to a term is equal to its frequency in the element. (This weighting scheme takes only term frequency into account.) Larger elements have more terms and terms with higher term frequency. Thus the probability of retrieval for larger elements is higher than that of the smaller elements irrespective of their relevance. In order to avoid this bias towards larger elements, we need to use a weighting scheme that takes element length into consideration and normalizes term weights based on it. Thus we use Singhal's *Lnu-ltu* weighting formula, which is based on slope & pivot. [10]

Pivot represents the average element length. The normalization is pivoted at pivot and tilted so that the elements on one side of pivot get larger normalization factors and the others get smaller normalization factors. The amount of *tilting* required is represented by *slope*. To reduce the difference between the probability of relevance and the probability of retrieval for all elements of different lengths, we also calculate slope. [5]

Slope and pivot are empirically determined constants. The pivot value varies from collection to collection and slope needs to be calculated accordingly. Table 2 presents the values of slope and pivot used for INEX 2007 document collection.

Element Level	Slope	Pivot
Article	0.04	120
Paragraph	0.12	18
All-element	0.12	38

**Table 2(a): Slope and Pivot values (Set 1- For old tags)**

Element Level	Slope	Pivot
Article	0.04	120
Paragraph	0.12	15
All-element	0.12	38

**Table 2(b): Slope and Pivot values (Set 2- For new tags)**

We convert the *nnn* weights on element vectors to *Lnu* weights using the formula given in Figure 4.

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

where, tf is term frequency (as in the *nnn* 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 weighting formula [5]**

The query vectors are weighted using the *ltu* weighting scheme as seen 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 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* weighting formula [5]**

### **Retrieval**

In this step, the *Lnu*-weighted element vectors are correlated with the *ltu*-weighted query vector using inner product, and a correlation score is assigned to each element. The elements are then sorted according to the correlation score which produces the final list of ranked elements. We specify the number of elements to be retrieved against every query.

### **3.2 Flex**

In this stage, we take highly correlating terminal node elements (from pre-Flex processing) as input and then construct the vectors for their parent elements. The output of this stage is a ranked list of elements.

### **Seeding Flex**

Retrieval against a set of terminal nodes (i.e., paragraph-and-mt) is required to seed Flex. The set of *n* top-ranked elements from this retrieval is used as the seed. The process of seeding is based on the assumption that if any element from a document has a strong correlation with the topic, there is a good probability that other elements, from the same

document, will correlate highly with the topic as well. Of particular interest are the following inputs to this process.

**1. Number of top-ranked elements (n):**

This parameter represents the number of elements fed to Flex (i.e., used to seed Flex). It also specifies the maximum number of trees that can be built by Flex on a per query basis.

**2. Document schemas (doc-trees):**

Doc-trees provide the structure/schema of all documents in document collection. They are produced using the same set of terminal node tags used for parsing. Each doc-tree represents a pre-order traversal of the corresponding document. Once the set of documents with highly correlating elements is determined, the trees for those documents are built using the document schema specified by the doc-trees.

**3. Docid-docpath mapping:**

The output of the retrieval is a set of elements represented by their Smart identifiers. In order for Flex to build the document trees, we need to map the Smart identifier of the element to its actual XPath. The Docid-docpath mapping is used to relate the XPath of every terminal element identified in the schema of the seeded document with its corresponding Smart identifier.

**Flex**

Flex reads the schema information of each seeded document. It then constructs the vectors for the non-terminal elements and computes the correlation of every element with the query vector. It outputs a ranked list of elements (with correlation scores) for every query.

### 3.3 Post-Flex Operations

In post-Flex operations, we first remove the mt elements from Flex results. The results are then converted into various subtasks. For each subtask, the results are converted to INEX-specified XML format and evaluated.

#### Filtering Flex Output

The output of Flex is a ranked list of elements. We first remove from this list all the nodes tagged with the mt tag, which is not a valid tag for the Wikipedia collection. (We use the mt tag in the parsing stage to include untagged text when element vectors are built. The importance of the mt tag is described in Section 3.4.) After removing mt elements, the output is further filtered based on the subtask. Outputs for all the subtasks are converted to INEX-specified XML format before evaluation.

1. **Focused Task:** The Focused task asks systems to find the most focused results that satisfy information need, without returning "overlapping" elements. That is, for a given topic, no retrieved element in the result set may contain text already contained in another element. After getting output from Flex, we remove overlap, thus converting it to the Focused result.

We have two different strategies for removing overlap. In first strategy, the element with the higher correlation score along the path is given priority (correlation score strategy). In the second strategy, we give precedence to the terminal node on a path, removing all other elements from that path (terminal node strategy). For the INEX 2007 Focused task, we use the terminal node overlap removal strategy, which has proved to be more effective. These two strategies are discussed in more detail in Chapter 4.

2. **Relevance-in-Context Task:** The aim of the Relevance-in-Context task is first to identify relevant articles (the fetching phase), and then to identify the relevant elements from those articles (the browsing phase). In the fetching phase, articles

should be ranked according to their topical relevance. Browsing should then provide a set of non-overlapping elements that cover the relevant information in the article. In our implementation of the Relevance-in-Contest task, we first perform article retrieval to identify highly correlating articles. These articles are used to seed Flex. All overlapping elements are then removed from Flex output leaving only positively correlating terminal nodes. The results are converted to INEX format. (See [1] for details.)

- 3. Best-in-Context Task:** The Best-in-Contest task asks systems to find that element from each relevant article that represents the best entry point (i.e., the location at which the user should start reading the article). We have several strategies for selecting the single best-entry-point from an article. The factors of interest in this task include correlation score, physical position and element tag. (See [6] for details.)

### **XPath expansion and validation**

The XPaths in our result files are not complete since some of the intermediate tags in the XPaths are left out. We expand the XPaths in the result file to add these intermediate tags (See [1] for detail). Our results are then validated using validation software provided by INEX. This software looks for the completeness of XPaths and checks for the presence of overlap in the result file. It reports an error for invalid XPaths or presence of overlap.

### **Evaluation**

The output for each of the subtasks is converted to the INEX-specified XML format. Evaluation measures used in INEX 2007 are quite different from those used in previous INEX competitions. (We no longer use *evalJ*, the INEX 2006 evaluation package.) For evaluation of 2007 tasks, INEX provides 3 Perl scripts (one for each subtask). Using these scripts, the XML results for the subtasks are evaluated against the relevance assessments.

### 3.4 Importance of Magic Text

The magic text tag is required because of the semi-structured nature of the document collection. Figure 6 shows the structure of a typical semi-structured Wikipedia document.

There is text present in the *section* which is not labeled with the *paragraph* tag. This text is called untagged text. Flex requires the presence of each discrete leaf node so that an exact copy of the parent element can be generated. Flex cannot generate the correct parent vector unless all the untagged text is included.

```
<article>
  <section>
    text1
    <p> text 2</p>
  </section>
</article>
```

**Figure 6: Typical semi-structured Wikipedia document**

We use magic text tags to deal with the issue of untagged text. In the parsing stage, whenever untagged text is encountered, we create a magic text tag (<mt>) and place the untagged text within it. Thus all untagged text within an element is combined and labeled as an <mt> element of the parent. Each article or section can have at most one magic text element [4]. Figure 7 shows the conversion of a semi-structured document into a structured document.

In all the stages of Flexible retrieval after parsing, a magic text element is treated in the same way as any other paragraph element. After getting the final list of ranked elements from Flex, all magic text elements are removed (recall that <mt> is not a valid tag and cannot be returned).

```
<article>
  <section>
    <mt> text1 </mt>
    <p> text2 </p>
  </section>
</article>
```

**Figure 7: Completely structured document**

## 4. Improving Focused Retrieval

In this chapter, the procedure for Focused retrieval using Flex is presented. It also explains in more detail our overlap removal strategy. We then discuss the use of a new set of terminal node tags (along with their slope/pivot values) in order to improve Focused retrieval results. Experimental results of Focused retrieval are given.

### 4.1 Focused Retrieval

The Focused Task requires retrieval systems to find the most focused results that satisfy the information need specified in the query. The main goal of this task is to return a ranked list of elements, in which no element overlaps any other element. The assumption of the Focused task is that smaller elements are more “focused” towards the query. Thus, if there is more than one element along a path with the same correlation score, the lower level (i.e., more precise) element is returned. Assuming that the retrieval system returns the highlighted text, the characters of text retrieved by the system are compared to the characters of text identified as relevant by the assessor for evaluation of Focused task results. [2] The following steps are implemented to produce results for the Focused task.

#### **Flexible Retrieval**

Flexible Retrieval refers to the task of retrieving specific elements in response to a query. We produce Flex results by following the steps explained in Chapter 3. The output of Flex is a ranked list of elements correlating with the query.

#### **Removing mts**

After producing a Flex result, we remove from it all nodes tagged with the magic text tag. (The importance of the magic text tag is explained in Section 3.9.)

### **Removing Overlap**

The Focused task asks systems to find the most focused results that satisfy information need, without returning "overlapping" elements. That is, for a given topic, no retrieved element in the result set may contain text already contained in another element. From the output of the previous step we remove overlap, thus producing a Focused result.

### **Conversion**

INEX specifies a format for submitting Focused task results. We convert our Focused result to INEX-specified XML format. The XML results are then validated using validation software provided by INEX. This software looks for the completeness of XPath expressions and checks for the presence of overlap in the result file. It reports an error for invalid XPath expressions or the presence of overlap.

### **Evaluation**

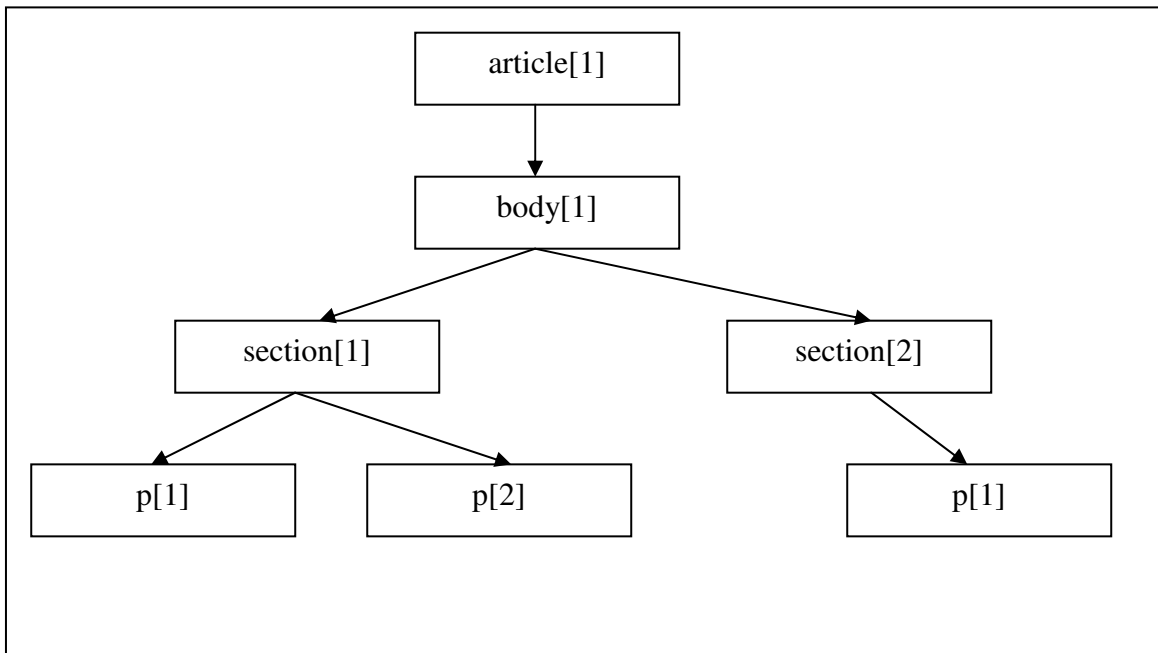
The XML result for the Focused subtask is then evaluated against the relevance assessment using INEX 2007 evaluation script.

## **4.2 Overlap Removal Strategies**

We have two different strategies for removing overlap, namely, the correlation score strategy and the terminal node strategy. The strategies are described below in Figures 8a and 8b. Figure 8a shows a sample result file containing XPath expressions from a single article. Figure 8b shows the tree structure represented by these XPath expressions.

article[1]/body[1]/section[1]	73.1168
...	
article[1]/body[1]/section[1]/p[1]	63.7365
...	
article[1]/body[1]/section[1]/p[2]	62.4254
...	
article[1]/body[1]/section[2]/p[1]	61.4285
...	
article[1]/body[1]/section[2]	60.5716

**Figure 8a: Excerpt from Flex result file showing XPathS from a single article (Article 211490)**



**Figure 8b: Tree structure represented by XPathS from Fig. 8a**

#### 4.2.1 Correlation Score Strategy:

In this strategy, the element with the highest correlation along a path is given priority. Figure 8a shows that *article[1]/body[1]/section[1]* has the highest correlation along the first path, and *article[1]/body[1]/section[2]/p[1]* has the highest correlation along the second path. So using the correlation score strategy, the following XPathS will be selected.

*article[1]/body[1]/section[1]*

*article[1]/body[1]/section[2]/p[1]*

All other XPathS will be removed.

#### 4.2.2 Terminal Node Strategy:

In this strategy, we always give precedence to the terminal node on a path (in effect, removing all other elements from that path). Figure 8b shows that *article[1]/body[1]/section[1]/p[1]* and *article[1]/body[1]/section[1]/p[2]* are the terminal nodes along the first path whereas *article[1]/body[1]/section[2]/p[1]* is the terminal node along the second path. So after removing overlap using the terminal node strategy, the following XPathS will be selected.

*article[1]/body[1]/section[1]/p[1]*

*article[1]/body[1]/section[1]/p[2]*

*article[1]/body[1]/section[2]/p[1]*

In the Focused task experiments described in this thesis, we use the terminal node strategy for removing overlap. The Focused task requires systems to find the most focused elements. The correlation score strategy carries with it the possibility of selecting highly correlating elements which may not be the most focused. For example, *article[1]/body[1]/section[1]* will be selected as the most focused element from path 1 using the correlation score strategy. But *section[1]* may contain other paragraphs (or sections) that do not correlate with the query. Focused result evaluation considers the number of characters in each element from the Focused result file. So by selecting *section[1]*, the text from non-correlating paragraphs and sections is included in the

“focused” element, which reduces performance. By selecting the positively correlating terminal node on a path, we eliminate the possibility of returning non-correlating text and thus improve performance.

### **4.3 2007 Results**

In this section, we provide some experimental Focused task results for INEX 2007 queries. The results are generated by following the Flex steps described in Chapter 3. Overlap is removed from Flex results to convert them into Focused results. The Focused result contains a ranked list of the top-ranked 1500 elements for each query. These results are converted to INEX-specified XML format. The Focused results in XML format are then evaluated against INEX 2007 Relevance Assessment Version 4.

In the result tables given below,  $n$  represents the number of elements used to seed Flex. It also specifies the maximum number of trees built by Flex.  $iP[x.xx]$  represents interpolated precision at  $x.xx$  recall, and  $MAiP$  represents the Mean Average Interpolated Precision. INEX ranks the Focused results using the interpolated precision value at 0.01 recall ( $iP[0.01]$ ).

#### **4.3.1 Old Terminal Node Tags (Old Slope/Pivot)**

The following results are generated using our original (i.e., old set of) terminal node tags (see Table 1(a)) and the corresponding slope/pivot values (see Table 2(a)). We discuss these results in Section 4.4

**Task: Focused**

**Overlap Removal: Correlation score strategy**

<b>n</b>	<b>iP[0.00]</b>	<b>iP[0.01]</b>	<b>iP[0.05]</b>	<b>iP[0.10]</b>	<b>MAiP</b>
2000	0.4957	0.4466	0.3296	0.2604	0.0879
1000	0.5189	0.4481	0.3078	0.2372	0.0703
500	0.5175	0.4473	0.3097	0.2401	0.0782
250	0.5181	0.4469	0.3122	0.2413	0.0797
100	0.5173	0.4590	0.3154	0.2517	0.0801
50	0.5162	<b>0.4502</b>	0.3297	0.2519	0.0794
25	0.5136	0.4491	0.3270	0.2493	0.0772
10	0.5084	0.4414	0.3031	0.2397	0.0601
5	0.4886	0.4173	0.2569	0.2194	0.0516
1	0.4008	0.3121	0.2178	0.1565	0.0430

**Table 3: 2007 Focused results for old tags using correlation score strategy**

**Task: Focused**

**Overlap Removal: Terminal node strategy**

<b>n</b>	<b>iP[0.00]</b>	<b>iP[0.01]</b>	<b>iP[0.05]</b>	<b>iP[0.10]</b>	<b>MAiP</b>
2000	0.5075	0.4554	0.3390	0.2655	0.0883
1000	0.5259	0.4590	0.3182	0.2449	0.0769
500	0.5249	0.4583	0.3197	0.2482	0.0808
250	0.5251	0.4585	0.3228	0.2532	0.0837
100	0.5247	0.4586	0.3294	0.2638	0.0849
50	0.5222	0.4582	0.3390	0.2618	0.0846
25	0.5233	<b>0.4642</b>	0.3340	0.2530	0.0810
10	0.5160	0.4613	0.3182	0.2489	0.0694
5	0.5086	0.4336	0.2868	0.2395	0.0631
1	0.4008	0.3121	0.2178	0.1565	0.0430

**Table 4: 2007 Focused results for old tags using terminal node strategy**

### 4.3.2 New Terminal Node tags (New Slope/Pivot)

In the first step of flexible retrieval, parsing, we recognize a set of tags as the terminal node tags. Selecting this set of terminal node tags has a large impact on Flex results. After a careful analysis of the INEX 2007 relevance assessments, we identified elements more frequently marked by relevance assessors. To improve the Focused task results, we selected a new set of terminal node tags which were quite different from the set used previously. Section 3.1 presents tables for both Set 1(Old tags) and Set 2(New tags).

With a new set of terminal node tags, the average element length changes. To incorporate this change, we recalculate the value of pivot and modify the value of slope accordingly. Section 3.3 presents tables for Set 1(Old Slope/Pivot) and Set 2(New Slope/Pivot). We performed Focused retrieval using both sets of tags. Each set of tags carries with it its own values of slope and pivot. Results generated using this new set of terminal node tags are given below in tables 5 and 6.

#### Task: Focused

#### Overlap Removal: Correlation score strategy

n	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
2000	0.5436	0.4809	0.3167	0.2482	0.0699
1000	0.5531	<b>0.4814</b>	0.3197	0.2516	0.0678
500	0.5539	0.4793	0.3201	0.2538	0.0712
250	0.5521	0.4780	0.3176	0.2535	0.0737
100	0.5509	0.4767	0.3160	0.2521	0.0707
50	0.5498	0.4781	0.3214	0.2510	0.0699
25	0.5512	0.4774	0.3243	0.2545	0.0711
10	0.5334	0.4567	0.2913	0.2212	0.0668
5	0.5155	0.4286	0.2505	0.1872	0.0548
1	0.3897	0.2406	0.1368	0.1007	0.0224

**Table 5: 2007 Focused results for new terminal node tags using correlation score strategy**

## Task: Focused

### Overlap Removal: Terminal node strategy

n	iP[0.00]	iP[0.01]	iP[0.05]	iP[0.10]	MAiP
2000	0.5591	0.4995	0.3280	0.2557	0.0747
1000	0.5635	<b>0.5022</b>	0.3371	0.2660	0.0775
500	0.5633	0.4993	0.3360	0.2689	0.0817
250	0.5612	0.4977	0.3327	0.2685	0.0828
100	0.5605	0.4955	0.3310	0.2641	0.0810
50	0.5600	0.4958	0.3374	0.2614	0.0809
25	0.5609	0.4954	0.3449	0.2674	0.0816
10	0.5452	0.4701	0.3045	0.2320	0.0732
5	0.5271	0.4400	0.2657	0.1952	0.0609
1	0.4019	0.2574	0.1589	0.1272	0.0329

**Table 6: 2007 Focused results for new terminal node tags using terminal node strategy**

## 4.4 Analysis

INEX provides rankings for the Focused results of all participants using the interpolated precision value at 0.01 recall ( $iP[0.01]$ ). To analyze our results, we compared our best value at  $iP[0.01]$  with other participants and checked our rank for that value. We observed the difference in the various strategies by checking the INEX ranking of the corresponding results.

As seen in Table 3, our best value at  $iP[0.01]$  for old tags using the correlation score strategy is **0.4502**. Our rank with this result is **40**. Table 4 shows the best value at  $iP[0.01]$  for old tags using the terminal node strategy as **0.4642** which improves our rank to **32**. Similarly, our best value at  $iP[0.01]$  for the new tags using the correlation score strategy is **0.4814** as shown in Table 5. Our rank with this result is **21**. Table 6 shows the best value at  $iP[0.01]$  for new tags using the terminal node strategy is **0.5022**, which improves our rank to **11**. Considering the difference in the rankings, we can conclude that

the terminal node strategy works more effectively than the correlation score strategy for Focused retrieval.

Considering only the terminal node strategy, our Focused result for the old tags (as seen in Table 4) gives us a rank of **32**, whereas the results for the new tags (as seen in Table 6) yields a rank of **11**. This difference in the rankings shows the improvement in the Focused retrieval produced by using the new terminal node tags.

## **5. Incorporating Article Retrieval with Flexible Retrieval**

This chapter presents the details of an experiment performed to improve Focused retrieval results. In this experiment, we incorporate article retrieval with element retrieval. The goal is to improve on the results produced by Focused retrieval.

### **5.1 Importance of Article Retrieval**

The results of Focused retrieval are evaluated against the relevance assessments provided by INEX. Relevance assessments are produced by the manual assessment of document elements against the query. For every query, INEX retrieves a number of articles using its own TopX retrieval engine and forms a pool of these articles. The relevance assessor then highlights the text from articles in this pool which he/she finds relevant to the topic.

INEX provides a pool of 650 articles (on average) per query. But the INEX 2007 document collection contains 659,338 Wikipedia articles. Our system performs retrieval against the entire collection. So our result may contain elements from articles outside the pool of 650. These elements are considered non-relevant by relevance assessment, so inclusion of such elements in our result would lower the evaluation score.

In seeking to avoid this, we first perform an article retrieval to identify a set of articles correlating with each query. In flexible retrieval, only the positively correlating elements from these articles are retrieved. In this way, we try to decrease the number of unwanted elements in our Focused retrieval.

### **5.2 Method**

We first perform article retrieval to identify the set of top-ranked articles. Then we filter the input of Flex according to this article set. Flex is seeded only with the highly correlating elements from the articles in this set. (For different numbers of articles, this

method involves change in only one step of flexible retrieval.) The steps involved in this process are given below.

- 1. Article Retrieval** - We retrieve a ranked list of  $A$  articles. (Set 1)
- 2. Terminal Node Retrieval** - We retrieve a ranked list of elements. (Set 2)
- 3. Creating Seed Subset** – The seed subset contains only the elements (from Set 2) that belong to the articles present in the article list (Set 1).
- 4. Flex** – The seed subset generated in the above step is used as an input to Flex. For each query, Flex generates a ranked list of  $P$  top-ranked elements as output.

Focused results are produced by removing overlap using the terminal node strategy, converting to INEX-specified XML format, and then evaluating against INEX 2007 Relevance Assessment Version 4.

### **5.3 Results**

Following are the results produced by using the aforementioned method for the INEX 2007 queries. In these tables,  $A$  represents the number of top-ranked articles retrieved and  $P$  represents the number of top-ranked elements returned for each query. Evaluation is done in terms of interpolated precision at 0.01 recall ( $iP[0.01]$ ). The results for 2007 using both the old and new sets of terminal node tags are given below.

**Relevance Assessment 2007: Version 4**

**Terminal node tags: Old set**

**Slope/Pivot: Old values (Slope = 0.12, Pivot = 18)**

<b># of Articles (A)</b>	<b># Paras (P) = 100</b>	<b># Paras (P) = 250</b>	<b># Paras (P) = 500</b>	<b># Paras (P) = 750</b>	<b># Paras (P) = 1000</b>	<b># Paras (P) = 1250</b>	<b># Paras (P) = 1500</b>
25	0.4855	0.4757	0.4792	0.4803	0.4800	0.4800	0.4800
50	0.4689	0.4720	0.4854	0.4870	0.4880	0.4855	<b>0.4882</b>
100	0.4719	0.4741	0.4846	0.4733	0.4741	0.4684	0.4744
150	0.4761	0.4644	0.4696	0.4660	0.4598	0.4670	0.4604
200	0.4735	0.4639	0.4568	0.4685	0.4689	0.4611	0.4567
250	0.4756	0.4815	0.4724	0.4732	0.4725	0.4689	0.4589

**Table 7: 2007 Focused results for old tags based on article retrieval**

**Relevance Assessment 2007: Version 4**

**Terminal node tags: New set**

**Slope/Pivot: New values (Slope = 0.12, Pivot = 15)**

<b># of Articles (A)</b>	<b># Paras (P) = 100</b>	<b># Paras (P) = 250</b>	<b># Paras (P) = 500</b>	<b># Paras (P) = 750</b>	<b># Paras (P) = 1000</b>	<b># Paras (P) = 1250</b>	<b># Paras (P) = 1500</b>
25	0.4871	0.5228	<b>0.5266</b>	0.5262	0.5262	0.5262	0.5262
50	0.4948	0.5192	0.5250	0.5255	0.5260	0.5260	0.5260
100	0.4632	0.4966	0.5146	0.5176	0.5166	0.5167	0.5175
150	0.4631	0.4699	0.5130	0.5090	0.5147	0.5167	0.5152
200	0.4637	0.4659	0.4923	0.5020	0.5046	0.5110	0.5129
250	0.4580	0.4589	0.4818	0.4980	0.4952	0.5042	0.5057

**Table 8: 2007 Focused results for new tags based on article retrieval**

## 5.4 Analysis

The above results are generated by incorporating article retrieval with element retrieval. To analyze the improvement in Focused results after using article retrieval, we compared the above results with the results generated using element retrieval only.

### Old terminal node tags

As seen in Table 4, our best value at  $iP[0.01]$  for the old tags using element retrieval only is **0.4642**(Rank **32**). Table 7 shows that the best value at  $iP[0.01]$  using article retrieval along with element retrieval is **0.4882**, which improves our rank to **17**.

### New terminal node tags

Our best value at  $iP[0.01]$  for the new tags using element retrieval only is **0.5022**, as seen in Table 6 (Rank **11**). Table 8 shows that the best value at  $iP[0.01]$  using article retrieval along with element retrieval as **0.5266**, which improves our rank to **2**.

Considering the difference in the rankings for both old and new terminal node tags, we can conclude that for Focused retrieval, incorporating article retrieval with element retrieval works more effectively than using element retrieval only.

All the aforementioned results are generated for the INEX 2007 Focused Task. These results are evaluated using 2007 relevance assessments. The INEX 2008 relevance assessments are not yet available. So, the strategy used to generate best Focused results for 2007 is used to produce 2008 Focused results.

## 6. Future Work

In this chapter, we make a few suggestions for improvements and future work.

The entire setup used in the experimentations described in this thesis can be made automatic with a single script for flexible retrieval process. Different weighting schemes for article retrieval could be considered. We could also fine tune the set of terminal node tags (and slope/pivot values) in hopes of improving Focused retrieval results.

The results reported in this thesis are based on element retrieval. For the INEX 2008 Ad hoc track, results can also be presented in terms of correlating passages. In the future, the strategies described in this thesis could be extended for passages by developing a new algorithm for passage retrieval. We could compare passage retrieval against element retrieval. This comparison would help us to determine if passage retrieval is more effective than element retrieval.

## References

- [1] Bapat, S. *Improving Results for Focused and Relevance-in-context tasks*, MS Thesis, University of Minnesota, Duluth, 2008.  
<http://www.d.umn.edu/cs/thesis>.
- [2] Fuhr, Norbert; Kamps, Jaap; Lalmas, Mounia; Malik, Saadia; Trotman, Andrew. INEX 2007 Workshop Pre-proceedings.
- [3] Initiative for the Evaluation of XML Retrieval (INEX)  
<http://inex.is.informatik.uni-duisburg.de/2007>.
- [4] Kamat, N. *Impact of Untagged Text in Dynamic Element Retrieval*, MS Thesis, University of Minnesota, Duluth, 2005.  
<http://www.d.umn.edu/cs/thesis>.
- [5] Khanna, S. *Design and Implementation of a Flexible Retrieval System*, MS Thesis, University of Minnesota, Duluth, 2005.  
<http://www.d.umn.edu/cs/thesis>.
- [6] Mehta, S. *Finding the Best Entry Point*, MS Thesis, University of Minnesota, Duluth, 2008.  
<http://www.d.umn.edu/cs/thesis>.
- [7] Pehcevski, Jovan; Kamps, Jaap; Kazai, Gabriella; Lalmas, Mounia; Ogilvie, Paul; Piwowarski, Benjamin; Robertson, Stephen. INEX 2007 Evaluation Measures.
- [8] Salton, G. editor. *The SMART Retrieval System – Experiments in Automatic Documents Retrieval*, Prentice-Hall, Eaglewood Cliffs, NJ, 1971.
- [9] Salton, G; Wong, A; Yang, C. *A vector space model for information retrieval*. *JASIS*, 18(11):613-620, 1975.
- [10] Singhal, A; Buckley, C; Mitra, M. *Pivoted Document Length Normalization*. In Proc. of the 19<sup>th</sup> Annual International ACM SIGIR Conference (Zurich, 1996).